W3School Html & Css 教程

wizardforcel

Published
with GitBook



目錄

W3School HTML & CSS教程

来源：

- HTML教程
- CSS教程
- Firebug教程

整理：飞龙

HTML 基础

HTML 简介

实例

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

什么是 HTML？

HTML 是用来描述网页的一种语言。

- HTML 指的是超文本标记语言 (**H**yper **T**ext **M**arkup **L**anguage)
- HTML 不是一种编程语言，而是一种标记语言 (markup language)
- 标记语言是一套标记标签 (markup tag)
- HTML 使用标记标签来描述网页

HTML 标签

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

- HTML 标签是由尖括号包围的关键词，比如 <html>
- HTML 标签通常是成对出现的，比如 和
- 标签对中的第一个标签是开始标签，第二个标签是结束标签
- 开始和结束标签也被称为开放标签和闭合标签

HTML 文档 = 网页

- HTML 文档描述网页
- HTML 文档包含 *HTML* 标签和纯文本
- HTML 文档也被称为网页

Web 浏览器的作用是读取 HTML 文档，并以网页的形式显示出它们。浏览器不会显示 HTML 标签，而是使用标签来解释页面的内容：

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

例子解释

- <html> 与 </html> 之间的文本描述网页
- <body> 与 </body> 之间的文本是可见的页面内容
- <h1> 与 </h1> 之间的文本被显示为标题
- <p> 与 </p> 之间的文本被显示为段落

基本的 HTML 标签 - 四个实例

本章通过实例向您演示最常用的 **HTML** 标签。

提示：不要担心本章中您还没有学过的例子，您将在下面的章节中学到它们。

提示：学习 HTML 最好的方式就是边学边做实验。我们为您准备了很好的 HTML 编辑器。使用这个编辑器，您可以任意编辑一段 HTML 源码，然后单击 TIY 按钮来查看结果。

HTML 标题

HTML 标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

实例

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML 段落

HTML 段落是通过 <p> 标签进行定义的。

实例

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML 链接

HTML 链接是通过 <a> 标签进行定义的。

实例

```
<a href="http://www.w3school.com.cn">This is a link</a>
```

注释：在 href 属性中指定链接的地址。

（您将在本教程稍后的章节中学习更多有关属性的知识）。

HTML 图像

HTML 图像是通过 <img> 标签进行定义的。

实例

```
<img src="w3school.jpg" width="104" height="142" />
```

注释：图像的名称和尺寸是以属性的形式提供的。

HTML 元素

HTML 文档是由 **HTML** 元素定义的。

HTML 元素

HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

开始标签	元素内容	结束标签
`<p>`	This is a paragraph	`</p>`
`<a href="default.htm" >`	This is a link	`</a>`
` `		

注释：开始标签常被称为开放标签（opening tag），结束标签常称为闭合标签（closing tag）。

HTML 元素语法

- HTML 元素以开始标签起始
- HTML 元素以结束标签终止
- 元素的内容是开始标签与结束标签之间的内容
- 某些 HTML 元素具有空内容（*empty content*）
- 空元素在开始标签中进行关闭（以开始标签的结束而结束）
- 大多数 HTML 元素可拥有属性

提示：您将在本教程的下一章中学习更多有关属性的内容。

嵌套的 HTML 元素

大多数 HTML 元素可以嵌套（可以包含其他 HTML 元素）。

HTML 文档由嵌套的 HTML 元素构成。

HTML 文档实例

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

上面的例子包含三个 HTML 元素。

HTML 实例解释

<p> 元素：

```
<p>This is my first paragraph.</p>
```

这个 <p> 元素定义了 HTML 文档中的一个段落。

这个元素拥有一个开始标签 <p>，以及一个结束标签 </p>。

元素内容是：This is my first paragraph。

<body> 元素：

```
<body>
<p>This is my first paragraph.</p>
</body>
```

<body> 元素定义了 HTML 文档的主体。

这个元素拥有一个开始标签 <body>，以及一个结束标签 </body>。

元素内容是另一个 HTML 元素（p 元素）。

<html> 元素：

```
<html>

<body>
<p>This is my first paragraph.</p>
</body>

</html>
```

<html> 元素定义了整个 HTML 文档。

这个元素拥有一个开始标签 <html>，以及一个结束标签 </html>。

元素内容是另一个 HTML 元素（body 元素）。

不要忘记结束标签

即使您忘记了使用结束标签，大多数浏览器也会正确地显示 HTML：

```
<p>This is a paragraph
<p>This is a paragraph
```

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生不可预料的结果或错误。

注释：未来的 HTML 版本不允许省略结束标签。

空的 HTML 元素

没有内容的 HTML 元素被称为空元素。空元素是在开始标签中关闭的。

 就是没有关闭标签的空元素（
 标签定义换行）。

在 XHTML、XML 以及未来版本的 HTML 中，所有元素都必须被关闭。

在开始标签中添加斜杠，比如
，是关闭空元素的正确方法，HTML、XHTML 和 XML 都接受这种方式。

即使
 在所有浏览器中都是有效的，但使用
 其实是更长远的保障。

HTML 提示：使用小写标签

HTML 标签对大小写不敏感：<P> 等同于 <p>。许多网站都使用大写的 HTML 标签。

W3School 使用的是小写标签，因为万维网联盟（W3C）在 HTML 4 中推荐使用小写，而在未来 (X)HTML 版本中强制使用小写。

HTML 属性

属性为 **HTML** 元素提供附加信息。

HTML 属性

HTML 标签可以拥有属性。属性提供了有关 HTML 元素的更多的信息。

属性总是以名称/值对的形式出现，比如：*name="value"*。

属性总是在 HTML 元素的开始标签中规定。

属性实例

HTML 链接由 <a> 标签定义。链接的地址在 href 属性中指定：

```
<a href="http://www.w3school.com.cn">This is a link</a>
```

更多 HTML 属性实例

属性例子 1:

<h1> 定义标题的开始。

<h1 align="center"> 拥有关于对齐方式的附加信息。

TIY：居中排列标题

属性例子 2:

<body> 定义 HTML 文档的主体。

<body bgcolor="yellow"> 拥有关于背景颜色的附加信息。

TIY：背景颜色

属性例子 3:

<table> 定义 HTML 表格。（您将在稍后的章节学习到更多有关 HTML 表格的内容）

<table border="1"> 拥有关于表格边框的附加信息。

HTML 提示：使用小写属性

属性和属性值对大小写不敏感。

不过，万维网联盟在其 HTML 4 推荐标准中推荐小写的属性/属性值。

而新版本的 (X)HTML 要求使用小写属性。

始终为属性值加引号

属性值应该始终被包括在引号内。双引号是最常用的，不过使用单引号也没有问题。

在某些个别的情况下，比如属性值本身就含有双引号，那么您必须使用单引号，例如：

```
name='Bill "HelloWorld" Gates'
```

HTML 属性参考手册

我们的完整的 HTML 参考手册提供了每个 HTML 元素可使用的合法属性的完整列表：

完整的 HTML 参考手册

下面列出了适用于大多数 HTML 元素的属性：

属性	值	描述
class	*classname*	规定元素的类名（classname）
id	*id*	规定元素的唯一 id
style	*style_definition*	规定元素的行内样式（inline style）
title	*text*	规定元素的额外信息（可在工具提示中显示）

如需更多关于标准属性的信息，请访问：

HTML 标准属性参考手册

HTML 标题

在 **HTML** 文档中，标题很重要。

HTML 标题

标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

<h1> 定义最大的标题。<h6> 定义最小的标题。

实例

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

注释：浏览器会自动地在标题的前后添加空行。

注释：默认情况下，HTML 会自动地在块级元素前后添加一个额外的空行，比如段落、标题元素前后。

标题很重要

请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号的文本而使用标题。

搜索引擎使用标题为您的网页的结构和内容编制索引。

因为用户可以通过标题来快速浏览您的网页，所以用标题来呈现文档结构是很重要的。

应该将 h1 用作主标题（最重要的），其后是 h2（次重要的），再其次是 h3，以此类推。

HTML 水平线

<hr /> 标签在 HTML 页面中创建水平线。

hr 元素可用于分隔内容。

实例

```
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
```

提示：使用水平线 (<hr> 标签) 来分隔文章中的小节是一个办法（但并不是唯一的办法）。

HTML 注释

可以将注释插入 HTML 代码中，这样可以提高其可读性，使代码更易被人理解。浏览器会忽略注释，也不会显示它们。

注释是这样写的：

实例

```
<!-- This is a comment -->
```

注释：开始括号之后（左边的括号）需要紧跟一个叹号，结束括号之前（右边的括号）不需要。

提示：合理地使用注释可以对未来的代码编辑工作产生帮助。

HTML 提示 - 如何查看源代码

您一定曾经在看到某个网页时惊叹道 "WOW! 这是如何实现的？"

如果您想找到其中的奥秘，只需要单击右键，然后选择"查看源文件"（IE）或"查看页面源代码"（Firefox），其他浏览器的做法也是类似的。这么做会打开一个包含页面 HTML 代码的窗口。

来自本页的实例

标题

隐藏的注释

水平线

HTML 标签参考手册

W3School 的标签参考手册提供了有关这些标题及其属性的更多信息。

您将在本教程下面的章节中学到更多有关 HTML 标签和属性的知识。

标签	描述
`<html>`	定义 HTML 文档。
`<body>`	定义文档的主体。
`<h1>` to `<h6>`	定义 HTML 标题
`<hr>`	定义水平线。
`<!-->`	定义注释。

HTML 段落

可以把 **HTML** 文档分割为若干段落。

HTML 段落

段落是通过 <p> 标签定义的。

实例

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

注释：浏览器会自动地在段落的前后添加空行。（<p> 是块级元素）

提示：使用空的段落标记 <p></p> 去插入一个空行是个坏习惯。用
 标签代替它！（但是不要用
 标签去创建列表。不要着急，您将在稍后的篇幅学习到 HTML 列表。）

不要忘记结束标签

即使忘了使用结束标签，大多数浏览器也会正确地将 HTML 显示出来：

实例

```
<p>This is a paragraph
<p>This is another paragraph
```

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生意想不到的结果和错误。

注释：在未来的 HTML 版本中，不允许省略结束标签。

提示：通过结束标签来关闭 HTML 是一种经得起未来考验的 HTML 编写方法。清楚地标记某个元素在何处开始，并在何处结束，不论对您还是对浏览器来说，都会使代码更容易理解。

HTML 折行

如果您希望在不产生一个新段落的情况下进行换行（新行），请使用
 标签：

```
<p>This is<br />a para<br />graph with line breaks</p>
```


 元素是一个空的 HTML 元素。由于关闭标签没有任何意义，因此它没有结束标签。

 还是

您也许发现
 与
 很相似。

在 XHTML、XML 以及未来的 HTML 版本中，不允许使用没有结束标签（闭合标签）的 HTML 元素。

即使
 在所有浏览器中的显示都没有问题，使用
 也是更长远的保障。

HTML 输出 - 有用的提示

我们无法确定 HTML 被显示的确切效果。屏幕的大小，以及对窗口的调整都可能导致不同的结果。

对于 HTML，您无法通过在 HTML 代码中添加额外的空格或换行来改变输出的效果。

当显示页面时，浏览器会移除源代码中多余的空格和空行。所有连续的空格或空行都会被算作一个空格。需要注意的是，HTML 代码中的所有连续的空行（换行）也被显示为一个空格。

（这个例子演示了一些 HTML 格式化方面的问题）

来自本页的实例

HTML 段落

换行

在 HTML 代码中的排版一首唐诗

更多实例

更多段落

HTML 标签参考手册

W3School 的标签参考手册提供了有关 HTML 元素及其属性的更多信息。

标签	描述
`<p>`	定义段落。
` `	插入单个折行（换行）。

HTML 文本格式化

HTML 可定义很多供格式化输出的元素，比如粗体和斜体字。

下面有很多例子，您可以亲自试试：

HTML 文本格式化实例

文本格式化

```
&lt;html&gt;

&lt;body&gt;

&lt;b&gt;This text is bold&lt;/b&gt;

&lt;br /&gt;

&lt;strong&gt;This text is strong&lt;/strong&gt;

&lt;br /&gt;

&lt;big&gt;This text is big&lt;/big&gt;

&lt;br /&gt;

&lt;em&gt;This text is emphasized&lt;/em&gt;

&lt;br /&gt;

&lt;i&gt;This text is italic&lt;/i&gt;

&lt;br /&gt;

&lt;small&gt;This text is small&lt;/small&gt;

&lt;br /&gt;

This text contains
&lt;sub&gt;subscript&lt;/sub&gt;

&lt;br /&gt;

This text contains
&lt;sup&gt;superscript&lt;/sup&gt;

&lt;/body&gt;
&lt;/html&gt;
```

预格式文本

```
&lt;html&gt;

&lt;body&gt;

&lt;pre&gt;
这是
预格式文本。
它保留了      空格
和换行。
&lt;/pre&gt;

&lt;p&gt;pre 标签很适合显示计算机代码：&lt;/p&gt;

&lt;pre&gt;
for i = 1 to 10
     print i
next i
&lt;/pre&gt;

&lt;/body&gt;
&lt;/html&gt;
```

“计算机输出”标签

```
&lt;html&gt;

&lt;body&gt;

&lt;code&gt;Computer code&lt;/code&gt;
&lt;br /&gt;
&lt;kbd&gt;Keyboard input&lt;/kbd&gt;
&lt;br /&gt;
&lt;tt&gt;Teletype text&lt;/tt&gt;
&lt;br /&gt;
&lt;samp&gt;Sample text&lt;/samp&gt;
&lt;br /&gt;
&lt;var&gt;Computer variable&lt;/var&gt;
&lt;br /&gt;

&lt;p&gt;
&lt;b&gt;注释：&lt;/b&gt;这些标签常用于显示计算机/编程代码。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

地址

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;address&gt;
Written by &lt;a href="mailto:webmaster@example.com"&gt;Donald Duc
Visit us at:&lt;br&gt;
Example.com&lt;br&gt;
Box 564, Disneyland&lt;br&gt;
USA
&lt;/address&gt;

&lt;/body&gt;
&lt;/html&gt;
```

缩写和首字母缩写

```
&lt;html&gt;

&lt;body&gt;

&lt;abbr title="etcetera"&gt;etc.&lt;/abbr&gt;
&lt;br /&gt;
&lt;acronym title="World Wide Web"&gt;WWW&lt;/acronym&gt;

&lt;p&gt;在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完

&lt;p&gt;仅对于 IE 5 中的 acronym 元素有效。&lt;/p&gt;

&lt;p&gt;对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文字方向

```
&lt;html&gt;

&lt;body&gt;

&lt;abbr title="etcetera"&gt;etc.&lt;/abbr&gt;
&lt;br /&gt;
&lt;acronym title="World Wide Web"&gt;WWW&lt;/acronym&gt;

&lt;p&gt;在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完

&lt;p&gt;仅对于 IE 5 中的 acronym 元素有效。&lt;/p&gt;

&lt;p&gt;对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

块引用

```
&lt;html&gt;

&lt;body&gt;

这是长的引用：
&lt;blockquote&gt;
这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引用。这是长的引
&lt;/blockquote&gt;

这是短的引用：
&lt;q&gt;
这是短的引用。
&lt;/q&gt;

&lt;p&gt;
使用 blockquote 元素的话，浏览器会插入换行和外边距，而 q 元素不会有任何特殊的
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

删除字效果和插入字效果

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;一打有 &lt;del&gt;二十&lt;/del&gt; &lt;ins&gt;十二&lt;/ins&g

&lt;p&gt;大多数浏览器会改写为删除文本和下划线文本。&lt;/p&gt;

&lt;p&gt;一些老式的浏览器会把删除文本和下划线文本显示为普通文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

如何查看 HTML 源码

您是否有过这样的经历，当你看到一个很棒的站点，你会很想知道开发人员是如何将它实现的？

你有没有看过一些网页，并且想知道它是如何做出来的呢？

要揭示一个网站的技术秘密，其实很简单。单击浏览器的“查看”菜单，选择“查看源文件”即可。随后你会看到一个弹出的窗口，窗口内就是实际的 HTML 代码。

文本格式化标签

标签	描述
<b>	定义粗体文本。
<big>	定义大号字。
<em>	定义着重文字。
<i>	定义斜体字。
<small>	定义小号字。
<strong>	定义加重语气。
<sub>	定义下标字。
<sup>	定义上标字。
<ins>	定义插入字。
<del>	定义删除字。
<s>	不赞成使用。使用 <del> 代替。
<strike>	不赞成使用。使用 <del> 代替。
<u>	不赞成使用。使用样式（style）代替。

“计算机输出”标签

标签	描述
<code>	定义计算机代码。
<kbd>	定义键盘码。
<samp>	定义计算机代码样本。
<tt>	定义打字机代码。
<var>	定义变量。
<pre>	定义预格式文本。
<listing>	不赞成使用。使用 <pre> 代替。
<plaintext>	不赞成使用。使用 <pre> 代替。
<xmp>	不赞成使用。使用 <pre> 代替。

引用、引用和术语定义

标签	描述
`<abbr>`	定义缩写。
`<acronym>`	定义首字母缩写。
`<address>`	定义地址。
`<bdo>`	定义文字方向。
`<blockquote>`	定义长的引用。
`<q>`	定义短的引用语。
`<cite>`	定义引用、引证。
`<dfn>`	定义一个定义项目。

延伸阅读：

改变文本的外观和含义

HTML 编辑器

使用 Notepad 或 TextEdit 来编写 HTML

可以使用专业的 HTML 编辑器来编辑 HTML：

- Adobe Dreamweaver
- Microsoft Expression Web
- CoffeeCup HTML Editor

不过，我们同时推荐使用文本编辑器来学习 HTML，比如 Notepad (PC) 或 TextEdit (Mac)。我们相信，使用一款简单的文本编辑器是学习 HTML 的好方法。

通过记事本，依照以下四步来创建您的第一张网页。

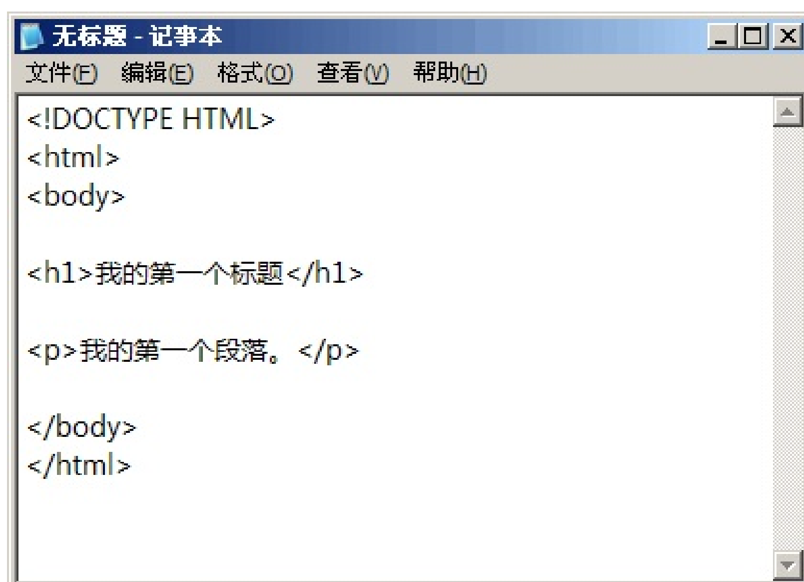
步骤一：启动记事本

如何启动记事本：

开始
所有程序
附件
记事本

步骤二：用记事本来编辑 HTML

在记事本中键入 HTML 代码：



步骤三：保存 HTML

在记事本的文件菜单选择“另存为”。

当您保存 HTML 文件时，既可以使用 .htm 也可以使用 .html 扩展名。两者没有区别，完全根据您的喜好。

在一个容易记忆的文件夹中保存这个文件，比如 w3school。

步骤四：在浏览器中运行这个 HTML 文件

启动您的浏览器，然后选择“文件”菜单的“打开文件”命令，或者直接在文件夹中双击您的 HTML 文件。

结果应该类似这样：



HTML CSS

通过使用 **HTML4.0**，所有的格式化代码均可移出 **HTML** 文档，然后移入一个独立的样式表。

实例

HTML中的样式

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
h1 {color: red}
p {color: blue}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h1&gt;header 1&lt;/h1&gt;
&lt;p&gt;A paragraph.&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

没有下划线的链接

```
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;a href="/example/html/lastpage.html" style="text-decoration:nor
这是一个链接！
&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

链接到一个外部样式表

```
&lt;html&gt;

&lt;head&gt;
&lt;link rel="stylesheet" type="text/css" href="/html/csstest1.css"
&lt;/head&gt;

&lt;body&gt;
&lt;h1&gt;我通过外部样式表进行格式化。&lt;/h1&gt;
&lt;p&gt;我也一样！&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

如何使用样式

当浏览器读到一个样式表，它就会按照这个样式表来对文档进行格式化。有以下三种方式来插入样式表：

外部样式表

当样式需要被应用到很多页面的时候，外部样式表将是理想的选择。使用外部样式表，你就可以通过更改一个文件来改变整个站点的外观。

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

内部样式表

当单个文件需要特别样式时，就可以使用内部样式表。你可以在 head 部分通过 <style> 标签定义内部样式表。

```
<head>

<style type="text/css">
body {background-color: red}
p {margin-left: 20px}
</style>
</head>
```

内联样式

当特殊的样式需要应用到个别元素时，就可以使用内联样式。 使用内联样式的方法是在相关的标签中使用样式属性。样式属性可以包含任何 CSS 属性。以下实例显示出如何改变段落的颜色和左外边距。

```
<p style="color: red; margin-left: 20px">
This is a paragraph
</p>
```

访问我们的 CSS 教程，学习更多有关样式的知识。

标签	描述
<style>	定义样式定义。
<link>	定义资源引用。
<div>	定义文档中的节或区域（块级）。
<span>	定义文档中的行内的小块或区域。
<font>	规定文本的字体、字体尺寸、字体颜色。不赞成使用。请使用样式。
<basefont>	定义基准字体。不赞成使用。请使用样式。
<center>	对文本进行水平居中。不赞成使用。请使用样式。

HTML 链接

HTML 使用超级链接与网络上的另一个文档相连。

几乎可以在所有的网页中找到链接。点击链接可以从一张页面跳转到另一张页面。

实例

创建超级链接

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="/index.html"&gt;本文本&lt;/a&gt; 是一个指向本网站中的一个页

&lt;p&gt;&lt;a href="http://www.microsoft.com/"&gt;本文本&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

将图像作为链接

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;
您也可以使用图像来作链接：
&lt;a href="/example/html/lastpage.html"&gt;
&lt;img border="0" src="/i/eg_buttonnext.gif" /&gt;
&lt;/a&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例）

HTML 超链接（链接）

超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，您可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当您把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：

1. 通过使用 href 属性 - 创建指向另一个文档的链接
2. 通过使用 name 属性 - 创建文档内的书签

延伸阅读：什么是超文本？

HTML 链接语法

链接的 HTML 代码很简单。它类似这样：

```
<a href="url">Link text</a>
```

href 属性规定链接的目标。

开始标签和结束标签之间的文字被作为超级链接来显示。

实例

```
<a href="http://www.w3school.com.cn/">Visit W3School</a>
```

上面这行代码显示为：Visit W3School

点击这个超链接会把用户带到 W3School 的首页。

提示："链接文本" 不必一定是文本。图片或其他 HTML 元素都可以成为链接。

HTML 链接 - target 属性

使用 Target 属性，你可以定义被链接的文档在何处显示。

下面的这行会在新窗口打开文档：

```
<a href="http://www.w3school.com.cn/" target="_blank">Visit W3Schoo
```

HTML 链接 - name 属性

name 属性规定锚（anchor）的名称。

您可以使用 name 属性创建 HTML 页面中的书签。

书签不会以任何特殊方式显示，它对读者是不可见的。

当使用命名锚（named anchors）时，我们可以创建直接跳至该命名锚（比如页面中某个小节）的链接，这样使用者就无需不停地滚动页面来寻找他们需要的信息了。

命名锚的语法：

```
<a name="_label_">锚（显示在页面上的文本）</a>
```

提示：锚的名称可以是任何你喜欢的名字。

提示：您可以使用 id 属性来替代 name 属性，命名锚同样有效。

实例

首先，我们在 HTML 文档中对锚进行命名（创建一个书签）：

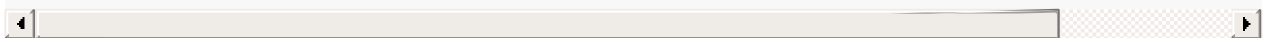
```
<a name="tips">基本的注意事项 - 有用的提示</a>
```

然后，我们在同一个文档中创建指向该锚的链接：

```
<a href="#tips">有用的提示</a>
```

您也可以在其他页面中创建指向该锚的链接：

```
<a href="http://www.w3school.com.cn/html/html_links.asp#tips">有用的
```



在上面的代码中，我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了。

具体效果：有用的提示

基本的注意事项 - 有用的提示：

注释：请始终将正斜杠添加到子文件夹。假如这样书写链接：href="http://www.w3school.com.cn/html"，就会向服务器产生两次 HTTP 请求。这是因为服务器会添加正斜杠到这个地址，然后创建一个新的请求，就像这样：href="http://www.w3school.com.cn/html/"。

提示：命名锚经常用于在大型文档开始位置上创建目录。可以为每个章节赋予一个命名锚，然后把链接到这些锚的链接放到文档的上部。如果您经常访问百度百科，您会发现其中几乎每个词条都采用这样的导航方式。

提示：假如浏览器找不到已定义的命名锚，那么就会定位到文档的顶端。不会有错误发生。

更多实例

在新的浏览器窗口打开链接

```
&lt;html&gt;

&lt;body&gt;

&lt;a href="http://www.w3school.com.cn/" target="_blank"&gt;Visit W

&lt;p&gt;如果把链接的 target 属性设置为 "_blank"，该链接会在新窗口中打开。&

&lt;/body&gt;

&lt;/html&gt;
```

链接到同一个页面的不同位置

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="#C4"&gt;查看 Chapter 4。&lt;/a&gt;
&lt;/p&gt;

&lt;h2&gt;Chapter 1&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 2&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 3&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;&lt;a name="C4"&gt;Chapter 4&lt;/a&gt;&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 5&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;
```

```
&lt;h2&gt;Chapter 6&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 7&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 8&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 9&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 10&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 11&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 12&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 13&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 14&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 15&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 16&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;h2&gt;Chapter 17&lt;/h2&gt;
&lt;p&gt;This chapter explains ba bla bla&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

跳出框架

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;被锁在框架中了吗？&lt;/p&gt;

&lt;a href="/index.html"
target="_top"&gt;请点击这里！&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建电子邮件链接

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
这是邮件链接：
&lt;a href="mailto:someone@microsoft.com?subject=Hello%20again"&gt
&lt;/p&gt;

&lt;p&gt;
&lt;b&gt;注意：&lt;/b&gt;应该使用 %20 来替换单词之间的空格，这样浏览器就可
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建电子邮件链接 2

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
这是另一个 mailto 链接：
&lt;a href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.c
&lt;/p&gt;

&lt;p&gt;
&lt;b&gt;注意：&lt;/b&gt;应该使用 %20 来替换单词之间的空格，这样浏览器就可
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML 链接标签

标签	描述
<a>	定义锚。

延伸阅读

什么是超文本？

HTML 图像

通过使用 **HTML**，可以在文档中显示图像。

实例

插入图像

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
一幅图像：
&lt;img src="/i/eg_mouse.jpg" width="128" height="128" /&gt;
&lt;/p&gt;

&lt;p&gt;
一幅动画图像：
&lt;img src="/i/eg_cute.gif" width="50" height="50" /&gt;
&lt;/p&gt;

&lt;p&gt;请注意，插入动画图像的语法与插入普通图像的语法没有区别。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

从不同的位置插入图片

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
来自另一个文件夹的图像：
&lt;img src="/i/ct_netscape.jpg" /&gt;
&lt;/p&gt;

&lt;p&gt;
来自 W3School.com.cn 的图像：
&lt;img src="img/w3school_logo_white.gif" /&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

图像标签（<img>）和源属性（Src）

在 HTML 中，图像由 <img> 标签定义。

<img> 是空标签，意思是说，它只包含属性，并且没有闭合标签。

要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

定义图像的语法是：

```
<img src="url" />
```

URL 指存储图像的位置。如果名为 "boat.gif" 的图像位于 www.w3school.com.cn 的 images 目录中，那么其 URL 为 http://www.w3school.com.cn/images/boat.gif。

浏览器将图像显示在文档中图像标签出现的地方。如果你将图像标签置于两个段落之间，那么浏览器会首先显示第一个段落，然后显示图片，最后显示第二段。

替换文本属性（Alt）

alt 属性用来为图像定义一串预备的可替换的文本。替换文本属性的值是用户定义的。

```
<img src="boat.gif" alt="Big Boat">
```

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。此时，浏览器将显示这个替代性的文本而不是图像。为页面上的图像都加上替换文本属性是个好习惯，这样有助于更好的显示信息，并且对于那些使用纯文本浏览器的人来说是非常有用的。

基本的注意事项 - 有用的提示：

假如某个 HTML 文件包含十个图像，那么为了正确显示这个页面，需要加载 11 个文件。加载图片是需要时间的，所以我们的建议是：慎用图片。

更多实例

背景图片

```
&lt;html&gt;

&lt;body background="/i/eg_background.jpg"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

排列图片

```
&lt;html&gt;

&lt;body&gt;

&lt;h2&gt;未设置对齐方式的图像：&lt;/h2&gt;

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif"&gt; 在文本中&lt;/p&gt;

&lt;h2&gt;已设置对齐方式的图像：&lt;/h2&gt;

&lt;p&gt;图像 &lt;img src="/i/eg_cute.gif" align="bottom"&gt; 在文本

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif" align="middle"&gt; 在文

&lt;p&gt;图像 &lt;img src ="/i/eg_cute.gif" align="top"&gt; 在文本中

&lt;p&gt;请注意，bottom 对齐方式是默认的对齐方式。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

浮动图像

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;
&lt;img src ="/i/eg_cute.gif" align ="left"&gt;
带有图像的一个段落。图像的 align 属性设置为 "left"。图像将浮动到文本的左侧。
&lt;/p&gt;

&lt;p&gt;
&lt;img src ="/i/eg_cute.gif" align ="right"&gt;
带有图像的一个段落。图像的 align 属性设置为 "right"。图像将浮动到文本的右侧。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

调整图像尺寸

```
&lt;html&gt;

&lt;body&gt;

&lt;img src="/i/eg_mouse.jpg" width="50" height="50"&gt;

&lt;br /&gt;

&lt;img src="/i/eg_mouse.jpg" width="100" height="100"&gt;

&lt;br /&gt;

&lt;img src="/i/eg_mouse.jpg" width="200" height="200"&gt;

&lt;p&gt;通过改变 img 标签的 "height" 和 "width" 属性的值，您可以放大或缩

&lt;/body&gt;
&lt;/html&gt;
```

为图片显示替换文本

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;仅支持文本的浏览器无法显示图像，仅仅能够显示在图像的 "alt" 属性中指

&lt;p&gt;请注意，如果您把鼠标指针移动到图像上，大多数浏览器会显示 "alt" 文本。

&lt;img src="/i/eg_goleft.gif" alt="向左转" /&gt;

&lt;p&gt;如果无法显示图像，将显示 "alt" 属性中的文本：&lt;/p&gt;

&lt;img src="/i/eg_goleft123.gif" alt="向左转" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

制作图像链接

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;
您也可以把图像作为链接来使用：
&lt;a href="/example/html/lastpage.html"&gt;
&lt;img border="0" src="/i/eg_buttonnext.gif" /&gt;
&lt;/a&gt;
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建图像映射

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;请点击图像上的星球，把它们放大。&lt;/p&gt;

&lt;img
src="/i/eg_planets.jpg"
border="0" usemap="#planetmap"
alt="Planets" /&gt;

&lt;map name="planetmap" id="planetmap"&gt;

&lt;area
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
alt="Venus" /&gt;

&lt;area
shape="circle"
coords="129,161,10"
href ="/example/html/mercur.html"
target ="_blank"
alt="Mercury" /&gt;

&lt;area
shape="rect"
coords="0,0,110,260"
href ="/example/html/sun.html"
target ="_blank"
alt="Sun" /&gt;

&lt;/map&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;img 元素中的 "usemap" 属性引用 map 元

&lt;/body&gt;
&lt;/html&gt;
```

把图像转换为图像映射

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;请把鼠标移动到图像上，看一下状态栏的坐标如何变化。&lt;/p&gt;

&lt;a href="/example/html/html_ismap.html"&gt;
&lt;img src="/i/eg_planets.jpg" ismap /&gt;
&lt;/a&gt;

&lt;/body&gt;
&lt;/html&gt;
```

图像标签

标签	描述
`<img>`	定义图像。
`<map>`	定义图像地图。
`<area>`	定义图像地图中的可点击区域。

</table>

HTML 表格

你可以使用 **HTML** 创建表格。

实例

表格

```
&lt;html&gt;

&lt;body&gt;

&lt;p&gt;每个表格由 table 标签开始。&lt;/p&gt;
&lt;p&gt;每个表格行由 tr 标签开始。&lt;/p&gt;
&lt;p&gt;每个表格数据由 td 标签开始。&lt;/p&gt;

&lt;h4&gt;一列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;一行三列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;两行三列：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格边框

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;带有普通的边框：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有粗的边框：&lt;/h4&gt;
&lt;table border="8"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有很粗的边框：&lt;/h4&gt;
&lt;table border="15"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器显示如下：

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格和边框属性

如果不定义边框属性，表格将不显示边框。有时这很有用，但是大多数时候，我们希望显示边框。

使用边框属性来显示一个带有边框的表格：

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

表格的表头

表格的表头使用 <th> 标签进行定义。

大多数浏览器会把表头显示为粗体居中的文本：

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器显示如下：

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格中的空单元格

在一些浏览器中，没有内容的表格单元显示得不太好。如果某个单元格是空的（没有内容），浏览器可能无法显示出这个单元格的边框。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td></td>
<td>row 2, cell 2</td>
</tr>
</table>
```

浏览器可能会这样显示：

row 1, cell 1	row 1, cell 2
	row 2, cell 2

注意：这个空的单元格的边框没有被显示出来。为了避免这种情况，在空单元格中添加一个空格占位符，就可以将边框显示出来。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td> </td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器中显示如下：

row 1, cell 1	row 1, cell 2
row 2, cell 2	

更多实例

没有边框的表格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;这个表格没有边框：&lt;/h4&gt;
&lt;table&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;这个表格也没有边框：&lt;/h4&gt;
&lt;table border="0"&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格中的表头(Heading)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;表头：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;th&gt;电话&lt;/th&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;垂直的表头：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

空单元格

```
&lt;html&gt;

&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;Some text&lt;/td&gt;
  &lt;td&gt;Some text&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;&lt;/td&gt;
  &lt;td&gt;Some text&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;正如您看到的，其中一个单元没有边框。这是因为它是空的。在该单元中插入

&lt;p&gt;我们的技巧是在单元中插入一个 no-breaking 空格。&lt;/p&gt;

&lt;p&gt;no-breaking 空格是一个字符实体。如果您不清楚什么是字符实体，请阅读

&lt;p&gt;no-breaking 空格由和号开始 ("&")，然后是字符"nbsp"，并以分号结尾

&lt;/body&gt;
&lt;/html&gt;
```

带有标题的表格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;这个表格有一个标题，以及粗边框：&lt;/h4&gt;

&lt;table border="6"&gt;
&lt;caption&gt;我的标题&lt;/caption&gt;
&lt;tr&gt;
  &lt;td&gt;100&lt;/td&gt;
  &lt;td&gt;200&lt;/td&gt;
  &lt;td&gt;300&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;400&lt;/td&gt;
  &lt;td&gt;500&lt;/td&gt;
  &lt;td&gt;600&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
```

跨行或跨列的表格单元格

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;横跨两列的单元格：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;th colspan="2"&gt;电话&lt;/th&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;横跨两行的单元格：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;th&gt;姓名&lt;/th&gt;
  &lt;td&gt;Bill Gates&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;th rowspan="2"&gt;电话&lt;/th&gt;
  &lt;td&gt;555 77 854&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;555 77 855&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表格内的标签

```
&lt;html&gt;

&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;
   &lt;p&gt;这是一个段落。&lt;/p&gt;
   &lt;p&gt;这是另一个段落。&lt;/p&gt;
  &lt;/td&gt;
  &lt;td&gt;这个单元包含一个表格：
   &lt;table border="1"&gt;
   &lt;tr&gt;
     &lt;td&gt;A&lt;/td&gt;
     &lt;td&gt;B&lt;/td&gt;
   &lt;/tr&gt;
   &lt;tr&gt;
     &lt;td&gt;C&lt;/td&gt;
     &lt;td&gt;D&lt;/td&gt;
   &lt;/tr&gt;
   &lt;/table&gt;
  &lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;这个单元包含一个列表：
   &lt;ul&gt;
    &lt;li&gt;苹果&lt;/li&gt;
    &lt;li&gt;香蕉&lt;/li&gt;
    &lt;li&gt;菠萝&lt;/li&gt;
   &lt;/ul&gt;
  &lt;/td&gt;
  &lt;td&gt;HELLO&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单元格边距(Cell padding)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;没有 cellpadding：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有 cellpadding：&lt;/h4&gt;
&lt;table border="1"
cellpadding="10"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单元格间距(Cell spacing)

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;没有 cellspacing：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;带有 cellspacing：&lt;/h4&gt;
&lt;table border="1"
cellspacing="10"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

向表格添加背景颜色或背景图像

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;背景颜色：&lt;/h4&gt;
&lt;table border="1"
bgcolor="red"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;h4&gt;背景图像：&lt;/h4&gt;
&lt;table border="1"
background="/i/eg_bg_07.gif"&gt;
&lt;tr&gt;
  &lt;td&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td&gt;Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

向表格单元添加背景颜色或者背景图像

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;单元背景：&lt;/h4&gt;
&lt;table border="1"&gt;
&lt;tr&gt;
  &lt;td bgcolor="red"&gt;First&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
  &lt;td
  background="/i/eg_bg_07.gif"&gt;
  Second&lt;/td&gt;
  &lt;td&gt;Row&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

在表格单元中排列内容

```
&lt;html&gt;

&lt;body&gt;

&lt;table width="400" border="1"&gt;
 &lt;tr&gt;
  &lt;th align="left"&gt;消费项目....&lt;/th&gt;
  &lt;th align="right"&gt;一月&lt;/th&gt;
  &lt;th align="right"&gt;二月&lt;/th&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;衣服&lt;/td&gt;
  &lt;td align="right"&gt;$241.10&lt;/td&gt;
  &lt;td align="right"&gt;$50.20&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;化妆品&lt;/td&gt;
  &lt;td align="right"&gt;$30.00&lt;/td&gt;
  &lt;td align="right"&gt;$44.45&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;td align="left"&gt;食物&lt;/td&gt;
  &lt;td align="right"&gt;$730.40&lt;/td&gt;
  &lt;td align="right"&gt;$650.00&lt;/td&gt;
 &lt;/tr&gt;
 &lt;tr&gt;
  &lt;th align="left"&gt;总计&lt;/th&gt;
  &lt;th align="right"&gt;$1001.50&lt;/th&gt;
  &lt;th align="right"&gt;$744.65&lt;/th&gt;
 &lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

框架(frame)属性

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;frame 属性无法在 Internet Explorer

&lt;p&gt;Table with frame="box":&lt;/p&gt;
&lt;table frame="box"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
```

```
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="above":&lt;/p&gt;
&lt;table frame="above"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="below":&lt;/p&gt;
&lt;table frame="below"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="hsides":&lt;/p&gt;
&lt;table frame="hsides"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;p&gt;Table with frame="vsides":&lt;/p&gt;
&lt;table frame="vsides"&gt;
  &lt;tr&gt;
    &lt;th&gt;Month&lt;/th&gt;
    &lt;th&gt;Savings&lt;/th&gt;
  &lt;/tr&gt;
  &lt;tr&gt;
    &lt;td&gt;January&lt;/td&gt;
    &lt;td&gt;$100&lt;/td&gt;
  &lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```



表格标签

表格	描述
`<table>`	定义表格
`<caption>`	定义表格标题。
`<th>`	定义表格的表头。
`<tr>`	定义表格的行。
`<td>`	定义表格单元。
`<thead>`	定义表格的页眉。
`<tbody>`	定义表格的主体。
`<tfoot>`	定义表格的页脚。
`<col>`	定义用于表格列的属性。
`<colgroup>`	定义表格列的组。

HTML 列表

HTML 支持有序、无序和定义列表

实例

无序列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个无序列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

有序列表

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;ol&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
&lt;/ol&gt;

&lt;ol start="50"&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
  &lt;li&gt;茶&lt;/li&gt;
&lt;/ol&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

无序列表

无序列表是一个项目的列表，此列项目使用粗体圆点（典型的小黑圆圈）进行标记。

无序列表始于 <ul> 标签。每个列表项始于 。

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

浏览器显示如下：

- Coffee
- Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

有序列表

同样，有序列表也是一列项目，列表项目使用数字进行标记。

有序列表始于 <ol> 标签。每个列表项始于 <li> 标签。

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

浏览器显示如下：

1. Coffee
2. Milk

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

定义列表

自定义列表不仅仅是一列项目，而是项目及其注释的组合。

自定义列表以 <dl> 标签开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

浏览器显示如下：

Coffee

Milk

定义列表的列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

更多实例

不同类型的无序列表

```
&lt;html&gt;
&lt;body&gt;

&lt;h4&gt;Disc 项目符号列表：&lt;/h4&gt;
&lt;ul type="disc"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;h4&gt;Circle 项目符号列表：&lt;/h4&gt;
&lt;ul type="circle"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;h4&gt;Square 项目符号列表：&lt;/h4&gt;
&lt;ul type="square"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

不同类型的有序列表

```
&lt;html&gt;
&lt;body&gt;

&lt;h4&gt;数字列表：&lt;/h4&gt;
&lt;ol&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;字母列表：&lt;/h4&gt;
&lt;ol type="A"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;小写字母列表：&lt;/h4&gt;
&lt;ol type="a"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;罗马字母列表：&lt;/h4&gt;
&lt;ol type="I"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;h4&gt;小写罗马字母列表：&lt;/h4&gt;
&lt;ol type="i"&gt;
 &lt;li&gt;苹果&lt;/li&gt;
 &lt;li&gt;香蕉&lt;/li&gt;
 &lt;li&gt;柠檬&lt;/li&gt;
 &lt;li&gt;桔子&lt;/li&gt;
&lt;/ol&gt;

&lt;/body&gt;
&lt;/html&gt;
```

嵌套列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个嵌套列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶
    &lt;ul&gt;
    &lt;li&gt;红茶&lt;/li&gt;
    &lt;li&gt;绿茶&lt;/li&gt;
    &lt;/ul&gt;
  &lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

嵌套列表 2

```
&lt;html&gt;

&lt;body&gt;

&lt;h4&gt;一个嵌套列表：&lt;/h4&gt;
&lt;ul&gt;
  &lt;li&gt;咖啡&lt;/li&gt;
  &lt;li&gt;茶
    &lt;ul&gt;
    &lt;li&gt;红茶&lt;/li&gt;
    &lt;li&gt;绿茶
      &lt;ul&gt;
      &lt;li&gt;中国茶&lt;/li&gt;
      &lt;li&gt;非洲茶&lt;/li&gt;
      &lt;/ul&gt;
    &lt;/li&gt;
    &lt;/ul&gt;
  &lt;/li&gt;
  &lt;li&gt;牛奶&lt;/li&gt;
&lt;/ul&gt;

&lt;/body&gt;
&lt;/html&gt;
```

定义列表

```
&lt;html&gt;

&lt;body&gt;

&lt;h2&gt;一个定义列表：&lt;/h2&gt;

&lt;dl&gt;
   &lt;dt&gt;计算机&lt;/dt&gt;
   &lt;dd&gt;用来计算的仪器 ... ...&lt;/dd&gt;
   &lt;dt&gt;显示器&lt;/dt&gt;
   &lt;dd&gt;以视觉方式显示信息的装置 ... ...&lt;/dd&gt;
&lt;/dl&gt;

&lt;/body&gt;
&lt;/html&gt;
```

列表标签

标签	描述
	定义有序列表。
	定义无序列表。
	定义列表项。
<dl>	定义定义列表。
<dt>	定义定义项目。
<dd>	定义定义的描述。
<dir>	已废弃。使用 代替它。
<menu>	已废弃。使用 代替它。

HTML <div> 和 <span>

可以通过 **<div>** 和 **<span>** 将 **HTML** 元素组合起来。

HTML 块元素

大多数 HTML 元素被定义为块级元素或内联元素。

编者注："块级元素"译为 block level element，"内联元素"译为 inline element。

块级元素在浏览器显示时，通常会以新行来开始（和结束）。

例子：<h1>, <p>, <ul>, <table>

HTML 内联元素

内联元素在显示时通常不会以新行开始。

例子：<b>, <td>, <a>, <img>

HTML <div> 元素

HTML <div> 元素是块级元素，它是可用于组合其他 HTML 元素的容器。

<div> 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

如果与 CSS 一同使用，<div> 元素可用于对大的内容块设置样式属性。

<div> 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用 <table> 元素进行文档布局不是表格的正确用法。<table> 元素的作用是显示表格化的数据。

HTML <span> 元素

HTML <span> 元素是内联元素，可用作文本的容器。

<span> 元素也没有特定的含义。

当与 CSS 一同使用时，<span> 元素可用于为部分文本设置样式属性。

HTML 分组标签

标签	描述
`<div>`	定义文档中的分区或节（division/section）。
`<span>`	定义 span，用来组合文档中的行内元素。

HTML 布局

网页布局对改善网站的外观非常重要。

请慎重设计您的网页布局。

亲自试一试 - 实例

使用 <div> 元素的网页布局

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div#container{width:500px}
div#header {background-color:#99bbbb;}
div#menu {background-color:#ffff99;height:200px;width:150px;float:l
div#content {background-color:#EEEEEE;height:200px;width:350px;floa
div#footer {background-color:#99bbbb;clear:both;text-align:center;}
h1 {margin-bottom:0;}
h2 {margin-bottom:0;font-size:18px;}
ul {margin:0;}
li {list-style:none;}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;div id="container"&gt;

&lt;div id="header"&gt;
&lt;h1&gt;Main Title of Web Page&lt;/h1&gt;
&lt;/div&gt;

&lt;div id="menu"&gt;
&lt;h2&gt;Menu&lt;/h2&gt;
&lt;ul&gt;
&lt;li&gt;HTML&lt;/li&gt;
&lt;li&gt;CSS&lt;/li&gt;
&lt;li&gt;JavaScript&lt;/li&gt;
&lt;/ul&gt;
&lt;/div&gt;

&lt;div id="content"&gt;Content goes here&lt;/div&gt;

&lt;div id="footer"&gt;Copyright W3School.com.cn&lt;/div&gt;

&lt;/div&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用 <table> 元素的网页布局

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;table width="500" border="0"&gt;
&lt;tr&gt;
&lt;td colspan="2" style="background-color:#99bbbb;"&gt;
&lt;h1&gt;Main Title of Web Page&lt;/h1&gt;
&lt;/td&gt;
&lt;/tr&gt;

&lt;tr valign="top"&gt;
&lt;td style="background-color:#ffff99;width:100px;text-align:top;'
&lt;b&gt;Menu&lt;/b&gt;&lt;br /&gt;
HTML&lt;br /&gt;
CSS&lt;br /&gt;
JavaScript
&lt;/td&gt;
&lt;td style="background-color:#EEEEEE;height:200px;width:400px;tex
Content goes here&lt;/td&gt;
&lt;/tr&gt;

&lt;tr&gt;
&lt;td colspan="2" style="background-color:#99bbbb;text-align:cente
Copyright W3School.com.cn&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

网站布局

大多数网站会把内容安排到多个列中（就像杂志或报纸那样）。

可以使用 <div> 或者 <table> 元素来创建多列。CSS 用于对元素进行定位，或者为页面创建背景以及色彩丰富的外观。

提示：即使可以使用 HTML 表格来创建漂亮的布局，但设计表格的目的是呈现表格化数据 - 表格不是布局工具！

HTML 布局 - 使用 <div> 元素

div 元素是用于分组 HTML 元素的块级元素。

下面的例子使用五个 div 元素来创建多列布局：

实例

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
div#container{width:500px}
div#header {background-color:#99bbbb;}
div#menu {background-color:#ffff99; height:200px; width:100px; floa
div#content {background-color:#EEEEEE; height:200px; width:400px; f
div#footer {background-color:#99bbbb; clear:both; text-align:center
h1 {margin-bottom:0;}
h2 {margin-bottom:0; font-size:14px;}
ul {margin:0;}
li {list-style:none;}
</style>
</head>

<body>

<div id="container">

<div id="header">
<h1>Main Title of Web Page</h1>
</div>

<div id="menu">
<h2>Menu</h2>
<ul>
<li>HTML</li>
<li>CSS</li>
<li>JavaScript</li>
</ul>
</div>

<div id="content">Content goes here</div>

<div id="footer">Copyright W3School.com.cn</div>

</div>

</body>
</html>
```

上面的 HTML 代码会产生如下结果：



HTML 布局 - 使用表格

使用 HTML <table> 标签是创建布局的一种简单的方式。

可以使用 <div> 或者 <table> 元素来创建多列。CSS 用于对元素进行定位，或者为页面创建背景以及色彩丰富的外观。

提示：即使可以使用 HTML 表格来创建漂亮的布局，但设计表格的目的是呈现表格化数据 - 表格不是布局工具！

下面的例子使用三行两列的表格 - 第一和最后一行使用 colspan 属性来横跨两列：

实例

```
<!DOCTYPE html>
<html>
<body>

<table width="500" border="0">
<tr>
<td colspan="2" style="background-color:#99bbbb;">
<h1>Main Title of Web Page</h1>
</td>
</tr>

<tr valign="top">
<td style="background-color:#ffff99;width:100px;text-align:top;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript
</td>
<td style="background-color:#EEEEEE;height:200px;width:400px;text-a
Content goes here</td>
</tr>

<tr>
<td colspan="2" style="background-color:#99bbbb;text-align:center;'
Copyright W3School.com.cn</td>
</tr>
</table>

</body>
</html>
```

上面的 HTML 代码会产生以下结果：



HTML 布局 - 有用的提示

提示：使用 CSS 最大的好处是，如果把 CSS 代码存放到外部样式表中，那么站点会更易于维护。通过编辑单一的文件，就可以改变所有页面的布局。如需学习更多有关 CSS 的知识，请访问我们的 CSS 教程。

提示：由于创建高级的布局非常耗时，使用模板是一个快速的选项。通过搜索引擎可以找到很多免费的网站模板（您可以使用这些预先构建好的网站布局，并优化它们）。

HTML 布局标签

标签	描述
`<div>`	定义文档中的分区或节（division/section）。
`<span>`	定义 span，用来组合文档中的行内元素。

HTML 表单和输入

HTML 表单用于搜集不同类型的用户输入。

实例

文本域 (Text field)

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
名：
&lt;input type="text" name="firstname"&gt;
&lt;br /&gt;
姓：
&lt;input type="text" name="lastname"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

密码域

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
用户：
&lt;input type="text" name="user"&gt;
&lt;br /&gt;
密码：
&lt;input type="password" name="password"&gt;
&lt;/form&gt;
&lt;p&gt;
请注意，当您在密码域中键入字符时，浏览器将使用项目符号来代替这些字符。
&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

表单

表单是一个包含表单元素的区域。

表单元素是允许用户在表单中（比如：文本域、下拉列表、单选框、复选框等等）输入信息的元素。

表单使用表单标签（<form>）定义。

```
<form>
...
  input 元素
...
</form>
```

输入

多数情况下被用到的表单标签是输入标签（<input>）。输入类型是由类型属性（type）定义的。大多数经常被用到的输入类型如下：

文本域（**Text Fields**）

当用户要在表单中键入字母、数字等内容时，就会用到文本域。

```
<form>
First name:
<input type="text" name="firstname" />
<br />
Last name:
<input type="text" name="lastname" />
</form>
```

浏览器显示如下：

First name: Last name:

注意，表单本身并不可见。同时，在大多数浏览器中，文本域的缺省宽度是20个字符。

单选按钮（**Radio Buttons**）

当用户从若干给定的的选择中选取其一时，就会用到单选框。

```
<form>
<input type="radio" name="sex" value="male" /> Male
<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

浏览器显示如下：

○ Male ○ Female
注意，只能从中选取其一。

复选框（Checkboxes）

当用户需要从若干给定的选择中选取一个或若干选项时，就会用到复选框。

```
<form>
<input type="checkbox" name="bike" />
I have a bike
<br />
<input type="checkbox" name="car" />
I have a car
</form>
```

浏览器显示如下：

☐ I have a bike ☐ I have a car

表单的动作属性（Action）和确认按钮

当用户单击确认按钮时，表单的内容会被传送到另一个文件。表单的动作属性定义了目的文件的文件名。由动作属性定义的这个文件通常会对接收到的输入数据进行相关的处理。

```
<form name="input" action="html_form_action.asp" method="get">
Username:
<input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

浏览器显示如下：

Username: [ ] Submit

假如您在上面的文本框内键入几个字母，然后点击确认按钮，那么输入数据会传送到 "html_form_action.asp" 的页面。该页面将显示出输入的结果。

更多实例

复选框

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
我喜欢自行车：
&lt;input type="checkbox" name="Bike"&gt;
&lt;br /&gt;
我喜欢汽车：
&lt;input type="checkbox" name="Car"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

单选按钮

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
男性：
&lt;input type="radio" checked="checked" name="Sex" value="male" /&
&lt;br /&gt;
女性：
&lt;input type="radio" name="Sex" value="female" /&gt;
&lt;/form&gt;

&lt;p&gt;当用户点击一个单选按钮时，该按钮会变为选中状态，其他所有按钮会变为非

&lt;/body&gt;
&lt;/html&gt;
```

简单的下拉列表

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;select name="cars"&gt;
&lt;option value="volvo"&gt;Volvo&lt;/option&gt;
&lt;option value="saab"&gt;Saab&lt;/option&gt;
&lt;option value="fiat"&gt;Fiat&lt;/option&gt;
&lt;option value="audi"&gt;Audi&lt;/option&gt;
&lt;/select&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

另一个下拉列表

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;select name="cars"&gt;
&lt;option value="volvo"&gt;Volvo&lt;/option&gt;
&lt;option value="saab"&gt;Saab&lt;/option&gt;
&lt;option value="fiat" selected="selected"&gt;Fiat&lt;/option&gt;
&lt;option value="audi"&gt;Audi&lt;/option&gt;
&lt;/select&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文本域(Textarea)

```
&lt;html&gt;
&lt;body&gt;

&lt;p&gt;
This example cannot be edited
because our editor uses a textarea
for input,
and your browser does not allow
a textarea inside a textarea.
&lt;/p&gt;

&lt;textarea rows="10" cols="30"&gt;
The cat was playing in the garden.
```

创建按钮

```
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
&lt;input type="button" value="Hello world!"&gt;
&lt;/form&gt;

&lt;/body&gt;
&lt;/html&gt;
```

Fieldset around data

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;

&lt;body&gt;

&lt;form&gt;
  &lt;fieldset&gt;
    &lt;legend&gt;健康信息&lt;/legend&gt;
    身高：&lt;input type="text" /&gt;
    体重：&lt;input type="text" /&gt;
  &lt;/fieldset&gt;
&lt;/form&gt;

&lt;p&gt;如果表单周围没有边框，说明您的浏览器太老了。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

表单实例

带有输入框和确认按钮的表单

```
&lt;html&gt;
&lt;body&gt;

&lt;form action="/example/html/form_action.asp" method="get"&gt;
  &lt;p&gt;First name: &lt;input type="text" name="fname" /&gt;&lt
  &lt;p&gt;Last name: &lt;input type="text" name="lname" /&gt;&lt;
  &lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;请单击确认按钮，输入会发送到服务器上名为 "form_action.asp" 的页面

&lt;/body&gt;
&lt;/html&gt;
```

带有复选框的表单

```
&lt;html&gt;

&lt;body&gt;

&lt;form name="input" action="/html/html_form_action.asp" method="
I have a bike:
&lt;input type="checkbox" name="vehicle" value="Bike" checked="chec
&lt;br /&gt;
I have a car:
&lt;input type="checkbox" name="vehicle" value="Car" /&gt;
&lt;br /&gt;
I have an airplane:
&lt;input type="checkbox" name="vehicle" value="Airplane" /&gt;
&lt;br /&gt;&lt;br /&gt;
&lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action

&lt;/body&gt;
&lt;/html&gt;
```

带有单选按钮的表单

```
&lt;html&gt;

&lt;body&gt;

&lt;form name="input" action="/html/html_form_action.asp" method="
Male:
&lt;input type="radio" name="Sex" value="Male" checked="checked"&g
&lt;br /&gt;
Female:
&lt;input type="radio" name="Sex" value="Female"&gt;
&lt;br /&gt;
&lt;input type ="submit" value ="Submit"&gt;
&lt;/form&gt;

&lt;p&gt;如果您点击 "Submit" 按钮，您将把输入传送到名为 html_form_action

&lt;/body&gt;
&lt;/html&gt;
```

从表单发送电子邮件

```
&lt;html&gt;

&lt;body&gt;
&lt;form action="MAILTO:someone@w3school.com.cn" method="post" enc

&lt;h3&gt;这个表单会把电子邮件发送到 W3School。&lt;/h3&gt;
姓名：&lt;br /&gt;
&lt;input type="text" name="name" value="yourname" size="20"&gt;
&lt;br /&gt;
电邮：&lt;br /&gt;
&lt;input type="text" name="mail" value="yourmail" size="20"&gt;
&lt;br /&gt;
内容：&lt;br /&gt;
&lt;input type="text" name="comment" value="yourcomment" size="40"
&lt;br /&gt;&lt;br /&gt;
&lt;input type="submit" value="发送"&gt;
&lt;input type="reset" value="重置"&gt;

&lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;
```

表单标签

标签	描述
`<form>`	定义供用户输入的表单
`<input>`	定义输入域
`<textarea>`	定义文本域 (一个多行的输入控件)
`<label>`	定义一个控制的标签
`<fieldset>`	定义域
`<legend>`	定义域的标题
`<select>`	定义一个选择列表
`<optgroup>`	定义选项组
`<option>`	定义下拉列表中的选项
`<button>`	定义一个按钮
`<isindex>`	已废弃。由 `<input>` 代替。

HTML 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。

实例

垂直框架

```
&lt;html&gt;

&lt;frameset cols="25%,50%,25%"&gt;

  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

水平框架

```
&lt;html&gt;

&lt;frameset rows="25%,50%,25%"&gt;

  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。每份HTML文档称为一个框架，并且每个框架都独立于其他的框架。

使用框架的坏处：

- 开发人员必须同时跟踪更多的HTML文档

- 很难打印整张页面

框架结构标签（<frameset>）

编者注：frameset 标签也被某些文章和书籍译为框架集。

框架标签（Frame）

Frame 标签定义了放置在每个框架中的 HTML 文档。

在下面的这个例子中，我们设置了一个两列的框架集。第一列被设置为占据浏览器窗口的 25%。第二列被设置为占据浏览器窗口的 75%。HTML 文档 "frame_a.htm" 被置于第一个列中，而 HTML 文档 "frame_b.htm" 被置于第二个列中：

```
<frameset cols="25%,75%">
   <frame src="frame_a.htm">
   <frame src="frame_b.htm">
</frameset>
```

基本的注意事项 - 有用的提示：

假如一个框架有可见边框，用户可以拖动边框来改变它的大小。为了避免这种情况发生，可以在 <frame> 标签中加入：noresize="noresize"。

为不支持框架的浏览器添加 <noframes> 标签。

重要提示：不能将 <body></body> 标签与 <frameset></frameset> 标签同时使用！不过，假如你添加包含一段文本的 <noframes> 标签，就必须将这段文字嵌套于 <body></body> 标签内。（在下面的第一个实例中，可以查看它是如何实现的。）

更多实例

如何使用 <noframes> 标签

```
&lt;html&gt;

&lt;frameset cols="25%,50%,25%"&gt;
  &lt;frame src="/example/html/frame_a.html"&gt;
  &lt;frame src="/example/html/frame_b.html"&gt;
  &lt;frame src="/example/html/frame_c.html"&gt;

&lt;noframes&gt;
&lt;body&gt;您的浏览器无法处理框架！&lt;/body&gt;
&lt;/noframes&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

混合框架结构

```
&lt;html&gt;

&lt;frameset rows="50%,50%"&gt;

&lt;frame src="/example/html/frame_a.html"&gt;

&lt;frameset cols="25%,75%"&gt;
&lt;frame src="/example/html/frame_b.html"&gt;
&lt;frame src="/example/html/frame_c.html"&gt;
&lt;/frameset&gt;

&lt;/frameset&gt;

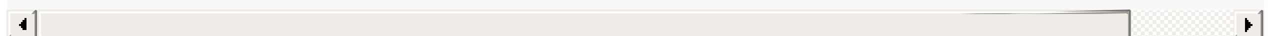
&lt;/html&gt;
```

含有 noresize="noresize" 属性的框架结构

```
&lt;html&gt;

&lt;frameset cols="50%,*,25%"&gt;
  &lt;frame src="/example/html/frame_a.html" noresize="noresize" /&
  &lt;frame src="/example/html/frame_b.html" /&gt;
  &lt;frame src="/example/html/frame_c.html" /&gt;
&lt;/frameset&gt;

&lt;/html&gt;
```



导航框架

```
&lt;html&gt;

&lt;frameset cols="120,*"&gt;

  &lt;frame src="/example/html/html_contents.html"&gt;
  &lt;frame src="/example/html/frame_a.html" name="showframe"&gt;

&lt;/frameset&gt;

&lt;/html&gt;
```

内联框架

```
&lt;html&gt;

&lt;body&gt;

&lt;iframe src="/i/eg_landscape.jpg"&gt;&lt;/iframe&gt;

&lt;p&gt;一些老的浏览器不支持 iframe。&lt;/p&gt;
&lt;p&gt;如果得不到支持，iframe 是不可见的。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

跳转至框架内的一个指定的节

```
&lt;html&gt;

&lt;frameset cols="20%,80%"&gt;

 &lt;frame src="/example/html/frame_a.html"&gt;
 &lt;frame src="/example/html/link.html#C10"&gt;

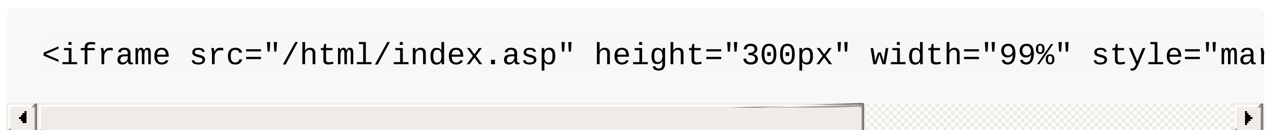
&lt;/frameset&gt;

&lt;/html&gt;
```

使用框架导航跳转至指定的节

HTML Iframe

iframe 用于在网页内显示网页。



```
<iframe src="/html/index.asp" height="300px" width="99%" style="ma
```

添加 iframe 的语法

```
<iframe src="_URL_"></iframe>
```

URL 指向隔离页面的位置。

Iframe - 设置高度和宽度

height 和 width 属性用于规定 iframe 的高度和宽度。

属性值的默认单位是像素，但也可以用百分比来设定（比如 "80%"）。

实例

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Iframe - 删除边框

frameborder 属性规定是否显示 iframe 周围的边框。

设置属性值为 "0" 就可以移除边框：

实例

```
<iframe src="demo_iframe.htm" frameborder="0"></iframe>
```

使用 iframe 作为链接的目标

iframe 可用作链接的目标（target）。

链接的 target 属性必须引用 iframe 的 name 属性：

实例

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3school.com.cn" target="iframe_a">W3School
```

HTML iframe 标签

标签	描述
`<iframe>`	定义内联的子窗口（框架）

HTML 背景

好的背景使站点看上去特别棒。

实例

搭配良好的背景和颜色

```
&lt;html&gt;

&lt;body bgcolor="#d0d0d0"&gt;

&lt;p&gt;
这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。
&lt;/p&gt;

&lt;p&gt;
这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

搭配得不好的背景和颜色

```
&lt;html&gt;

&lt;body bgcolor="#ffffff" text="yellow"&gt;

&lt;p&gt;
这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。这是段落。
&lt;/p&gt;

&lt;p&gt;
这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。这是另一个段落。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

（可以在本页底端找到更多实例。）

背景（Backgrounds）

<body> 拥有两个配置背景的标签。背景可以是颜色或者图像。

背景颜色（Bgcolor）

背景颜色属性将背景设置为某种颜色。属性值可以是十六进制数、RGB 值或颜色名。

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

以上的代码均将背景颜色设置为黑色。

背景（Background）

背景属性将背景设置为图像。属性值为图像的URL。如果图像尺寸小于浏览器窗口，那么图像将在整个浏览器窗口进行复制。

```
<body background="clouds.gif">
<body background="http://www.w3school.com.cn/clouds.gif">
```

URL可以是相对地址，如第一行代码。也可以使绝对地址，如第二行代码。

提示：如果你打算使用背景图片，你需要紧记一下几点：

- 背景图像是否增加了页面的加载时间。小贴士：图像文件不应超过 10k。
- 背景图像是否与页面中的其他图象搭配良好。
- 背景图像是否与页面中的文字颜色搭配良好。
- 图像在页面中平铺后，看上去还可以吗？
- 对文字的注意力被背景图像喧宾夺主了吗？

基本的注意事项 - 有用的提示：

<body> 标签中的背景颜色（bgcolor）、背景（background）和文本（text）属性在最新的 HTML 标准（HTML4 和 XHTML）中已被废弃。W3C 在他们的推荐标准中已删除这些属性。

应该使用层叠样式表（CSS）来定义 HTML 元素的布局和显示属性。

更多实例

可用性强的背景图像

```
&lt;html&gt;

&lt;body background="/i/eg_bg_06.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

可用性强的背景图像 2

```
&lt;html&gt;

&lt;body background="/i/eg_bg_04.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

可用性差的背景图像

```
&lt;html&gt;

&lt;body background="/i/eg_bg_01.gif"&gt;

&lt;h3&gt;图像背景&lt;/h3&gt;

&lt;p&gt;gif 和 jpg 文件均可用作 HTML 背景。&lt;/p&gt;

&lt;p&gt;如果图像小于页面，图像会进行重复。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML 颜色

颜色由红色、绿色、蓝色混合而成。

颜色值

颜色由一个十六进制符号来定义，这个符号由红色、绿色和蓝色的值组成（RGB）。

每种颜色的最小值是0（十六进制：#00）。最大值是255（十六进制：#FF）。

这个表格给出了由三种颜色混合而成的具体效果：

Color HEX	Color RGB
#000000	rgb(0,0,0)
#FF0000	rgb(255,0,0)
#00FF00	rgb(0,255,0)
#0000FF	rgb(0,0,255)
#FFFF00	rgb(255,255,0)
#00FFFF	rgb(0,255,255)
#FF00FF	rgb(255,0,255)
#C0C0C0	rgb(192,192,192)
#FFFFFF	rgb(255,255,255)

颜色名

大多数的浏览器都支持颜色名集合。

提示：仅仅有 16 种颜色名被 W3C 的 HTML4.0 标准所支持。它们是：aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow。

如果需要使用其它的颜色，需要使用十六进制的颜色值。

Color HEX	Color Name
#F0F8FF	AliceBlue
#FAEBD7	AntiqueWhite
#7FFFD4	Aquamarine
#000000	Black
#0000FF	Blue
#8A2BE2	BlueViolet
#A52A2A	Brown

Web安全色

数年以前，当大多数计算机仅支持 256 种颜色的时候，一系列 216 种 Web 安全色作为 Web 标准被建议使用。其中的原因是，微软和 Mac 操作系统使用了 40 种不同的保留的固定系统颜色（双方大约各使用 20 种）。

我们不确定如今这么做的意义有多大，因为越来越多的计算机有能力处理数百万种颜色，不过做选择还是你自己。

216 跨平台色

最初，216 跨平台 web 安全色被用来确保：当计算机使用 256 色调色板时，所有的计算机能够正确地显示所有的颜色。

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF

660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

HTML 颜色名

本页提供了被大多数浏览器支持的颜色名。

提示：仅有 16 种颜色名被 W3C 的 HTML 4.0 标准支持，它们是：aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow。

如果使用其它颜色的话，就应该使用十六进制的颜色值。

颜色名列表

单击一个颜色名或者 16 进制值，就可以查看与不同文字颜色搭配的背景颜色。

颜色名	十六进制颜色值
AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED

Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
Darkorange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DodgerBlue	#1E90FF
Feldspar	#D19275
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC

GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateBlue	#8470FF
LightSlateGray	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0

Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE

PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE

VioletRed	#D02090
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00
YellowGreen	#9ACD32

HTML 4.01 快速参考

来自 **W3School** 的 **HTML** 快速参考。可以打印它，以备日常使用。

HTML Basic Document

```
<html>
<head>
<title>Document name goes here</title>
</head>
<body>
Visible text goes here
</body>
</html>
```

Text Elements

```
<p>This is a paragraph</p>
<br> (line break)
<hr> (horizontal rule)
<pre>This text is preformatted</pre>
```

Logical Styles

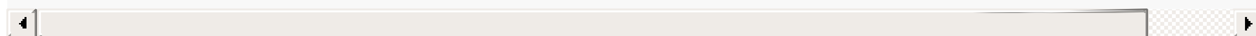
```
<em>This text is emphasized</em>
<strong>This text is strong</strong>
<code>This is some computer code</code>
```

Physical Styles

```
<b>This text is bold</b>
<i>This text is italic</i>
```

Links, Anchors, and Image Elements

```
<a href="http://www.example.com/">This is a Link</a>
<a href="http://www.example.com/"><img src="URL"
alt="Alternate Text"></a>
<a href="mailto:webmaster@example.com">Send e-mail</a>A named ancho
<a name="tips">Useful Tips Section</a>
<a href="#tips">Jump to the Useful Tips Section</a>
```



Unordered list

```
<ul>
<li>First item</li>
<li>Next item</li>
</ul>
```

Ordered list

```
<ol>
<li>First item</li>
<li>Next item</li>
</ol>
```

Definition list

```
<dl>
<dt>First term</dt>
<dd>Definition</dd>
<dt>Next term</dt>
<dd>Definition</dd>
</dl>
```

Tables

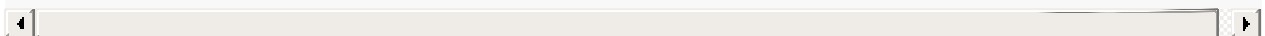
```
<table border="1">
<tr>
  <th>someheader</th>
  <th>someheader</th>
</tr>
<tr>
  <td>sometext</td>
  <td>sometext</td>
</tr>
</table>
```

Frames

```
<frameset cols="25%,75%">
  <frame src="page1.htm">
  <frame src="page2.htm">
</frameset>
```

Forms

```
<form action="http://www.example.com/test.asp" method="post/get">
<input type="text" name="lastname"
value="Nixon" size="30" maxlength="50">
<input type="password">
<input type="checkbox" checked="checked">
<input type="radio" checked="checked">
<input type="submit">
<input type="reset">
<input type="hidden">
<select>
<option>Apples
<option selected>Bananas
<option>Cherries
</select>
<textarea name="Comment" rows="60"
cols="20"></textarea>
</form>
```



Entities

```
&lt; is the same as <
&gt; is the same as >
&#169; is the same as ?
```

Other Elements

```
<!-- This is a comment -->
<blockquote>
Text quoted from some source.
</blockquote>
<address>
Address 1<br>
Address 2<br>
City<br>
</address>
```

Source : http://www.w3school.com.cn/html/html_quick.asp

HTML 高级

HTML <!DOCTYPE>

<!DOCTYPE> 声明帮助浏览器正确地显示网页。

<!DOCTYPE> 声明

Web 世界中存在许多不同的文档。只有了解文档的类型，浏览器才能正确地显示文档。

HTML 也有多个不同的版本，只有完全明白页面中使用的确切 HTML 版本，浏览器才能完全正确地显示出 HTML 页面。这就是 <!DOCTYPE> 的用处。

<!DOCTYPE> 不是 HTML 标签。它为浏览器提供一项信息（声明），即 HTML 是用什么版本编写的。

提示：W3School 即将升级为最新的 HTML5 文档类型。

实例

带有 HTML5 DOCTYPE 的 HTML 文档：

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

HTML 版本

从 Web 诞生早期至今，已经发展出多个 HTML 版本：

版本	年份
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML 1.0	2000
HTML5	2012
XHTML5	2013

常用的声明

HTML5

```
<!DOCTYPE html>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

如需完整的文档类型声明列表，请访问我们的 DOCTYPE 参考手册。

HTML 头部元素

亲自试一试 - 实例

文档的标题

```
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;

&lt;title&gt;标题不会显示在文档区&lt;/title&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;这段文本会显示出来。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

所有链接一个目标

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Content-Language" content="zh-cn" /&gt;

&lt;base target="_blank" /&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
&lt;a href="http://www.w3school.com.cn" target="_blank"&gt;这个连接&
&lt;/p&gt;

&lt;p&gt;
&lt;a href="http://www.w3school.com.cn"&gt;这个连接&lt;/a&gt; 也将在新
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

文档描述

```
&lt;html&gt;

&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23

&lt;meta name="author"
content="w3school.com.cn"&gt;

&lt;meta name="revised"
content="David Yang,8/1/07"&gt;

&lt;meta name="generator"
content="Dreamweaver 8.0en"&gt;

&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本文档的 meta 属性标识了创作者和编辑软件。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

文档关键词

```
&lt;html&gt;

&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23

&lt;meta name="description"
content="HTML examples"&gt;

&lt;meta name="keywords"
content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript"&gt;

&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本文档的 meta 属性描述了该文档和它的关键词。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

重定向用户

```
&lt;!DOCTYPE HTML&gt;
&lt;html&gt;
&lt;head&gt;
&lt;meta http-equiv="Content-Type" content="text/html; charset=gb23
&lt;meta http-equiv="Refresh" content="5;url=http://www.w3school.co
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;
对不起。我们已经搬家了。您的 URL 是 &lt;a href="http://www.w3school.com
&lt;/p&gt;

&lt;p&gt;您将在 5 秒内被重定向到新的地址。&lt;/p&gt;

&lt;p&gt;如果超过 5 秒后您仍然看到本消息，请点击上面的链接。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

HTML <head> 元素

<head> 元素是所有头部元素的容器。<head> 内的元素可包含脚本，指示浏览器在何处可以找到样式表，提供元信息，等等。

以下标签都可以添加到 head 部分：<title>、<base>、<link>、<meta>、<script> 以及 <style>。

HTML <title> 元素

<title> 标签定义文档的标题。

title 元素在所有 HTML/XHTML 文档中都是必需的。

title 元素能够：

- 定义浏览器工具栏中的标题
- 提供页面被添加到收藏夹时显示的标题
- 显示在搜索引擎结果中的页面标题

一个简化的 HTML 文档：

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

HTML <base> 元素

<base> 标签为页面上的所有链接规定默认地址或默认目标（target）：

```
<head>
<base href="http://www.w3school.com.cn/images/" />
<base target="_blank" />
</head>
```

HTML <link> 元素

<link> 标签定义文档与外部资源之间的关系。

<link> 标签最常用于连接样式表：

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

HTML <style> 元素

<style> 标签用于为 HTML 文档定义样式信息。

您可以在 style 元素内规定 HTML 元素在浏览器中呈现的样式：

```
<head>
<style type="text/css">
body {background-color:yellow}
p {color:blue}
</style>
</head>
```

HTML <meta> 元素

元数据（metadata）是关于数据的信息。

<meta> 标签提供关于 HTML 文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

典型的情况是，meta 元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。

<meta> 标签始终位于 head 元素中。

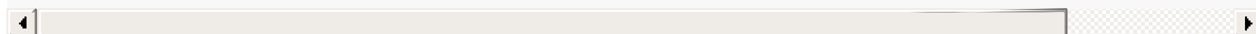
元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

针对搜索引擎的关键词

一些搜索引擎会利用 meta 元素的 name 和 content 属性来索引您的页面。

下面的 meta 元素定义页面的描述：

```
<meta name="description" content="Free Web tutorials on HTML, CSS,
```



下面的 meta 元素定义页面的关键词：

```
<meta name="keywords" content="HTML, CSS, XML" />
```

name 和 content 属性的作用是描述页面的内容。

HTML <script> 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

我们会在稍后的章节讲解 script 元素。

HTML 头部元素

标签	描述
`<head>`	定义关于文档的信息。
`<title>`	定义文档标题。
`<base>`	定义页面上所有链接的默认地址或默认目标。
`<link>`	定义文档与外部资源之间的关系。
`<meta>`	定义关于 HTML 文档的元数据。
`<script>`	定义客户端脚本。
`<style>`	定义文档的样式信息。

HTML 脚本

JavaScript 使 **HTML** 页面具有更强的动态和交互性。。

实例

插入一段脚本

```
&lt;html&gt;

&lt;body&gt;

&lt;script type="text/javascript"&gt;
document.write("&lt;h1&gt;Hello World!&lt;/h1&gt;")
&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;
```

使用 <noscript> 标签

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;

&lt;script type="text/javascript"&gt;
document.write("Hello World!")
&lt;/script&gt;
&lt;noscript&gt;Sorry, your browser does not support JavaScript!&lt

&lt;p&gt;不支持 JavaScript 的浏览器将显示 noscript 元素中的文本。&lt;/p&

&lt;/body&gt;
&lt;/html&gt;
```

HTML script 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

script 元素既可包含脚本语句，也可通过 src 属性指向外部脚本文件。

必需的 type 属性规定脚本的 MIME 类型。

JavaScript 最常用于图片操作、表单验证以及内容动态更新。

下面的脚本会向浏览器输出“Hello World!”：

```
<script type="text/javascript">
document.write("Hello World!")
</script>
```

提示：如果需要学习更多有关在 HTML 中编写脚本的知识，请访问我们的 JavaScript 教程。

<noscript> 标签

<noscript> 标签提供无法使用脚本时的替代内容，比方在浏览器禁用脚本时，或浏览器不支持客户端脚本时。

noscript 元素可包含普通 HTML 页面的 body 元素中能够找到的所有元素。

只有在浏览器不支持脚本或者禁用脚本时，才会显示 noscript 元素中的内容：

```
<script type="text/javascript">
document.write("Hello World!")
</script>
<noscript>Your browser does not support JavaScript!</noscript>
```

如何应付老式的浏览器

如果浏览器压根没法识别 <script> 标签，那么 <script> 标签所包含的内容将以文本方式显示在页面上。为了避免这种情况发生，你应该将脚本隐藏在注释标签当中。那些老的浏览器（无法识别 <script> 标签的浏览器）将忽略这些注释，所以不会将标签的内容显示到页面上。而那些新的浏览器将读懂这些脚本并执行它们，即使代码被嵌套在注释标签内。

实例

JavaScript:

```
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
```

VBScript:

```
<script type="text/vbscript">
<!--
document.write("Hello World!")
'-->
</script>
```

标签	描述
<script>	定义客户端脚本。
<noscript>	为不支持客户端脚本的浏览器定义替代内容。

HTML 字符实体

HTML 中的预留字符必须被替换为字符实体。

HTML 实体

在 HTML 中，某些字符是预留的。

在 HTML 中不能使用小于号（<）和大于号（>），这是因为浏览器会误认为它们是标签。

如果希望正确地显示预留字符，我们必须在 HTML 源代码中使用字符实体（character entities）。

字符实体类似这样：

```
&_entity_name_;

或者

&#_entity_number_;
```

如需显示小于号，我们必须这样写：< 或 <

提示：使用实体名而不是数字的好处是，名称易于记忆。不过坏处是，浏览器也许并不支持所有实体名称（对实体数字的支持却很好）。

不间断空格（**non-breaking space**）

HTML 中的常用字符实体是不间断空格( )。

浏览器总是会截短 HTML 页面中的空格。如果您在文本中写 10 个空格，在显示该页面之前，浏览器会删除它们中的 9 个。如需在页面中增加空格的数量，您需要使用   字符实体。

HTML 实例示例

用 HTML 实体符号做实验：

HTML 中有用的字符实体

注释：实体名称对大小写敏感！

显示结果	描述	实体名称	实体编号
	空格		
<	小于号	<	<
>	大于号	>	>
&	和号	&	&
"	引号	"	"
'	撇号	' (IE不支持)	'
¢	分	¢	¢
£	镑	£	£
¥	日圆	¥	¥
€	欧元	€	€
§	小节	§	§
©	版权	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

如需完整的实体符号参考，请访问我们的 HTML 实体符号参考手册。

HTML 统一资源定位器

URL 也被称为网址。

URL 可以由单词组成，比如 “w3school.com.cn”，或者是因特网协议（IP）地址：192.168.1.253。大多数人在网上冲浪时，会键入网址的域名，因为名称比数字容易记忆。

URL - Uniform Resource Locator

当您点击 HTML 页面中的某个链接时，对应的 <a> 标签指向万维网上的一个地址。

统一资源定位器（URL）用于定位万维网上的文档（或其他数据）。

网址，比如 http://www.w3school.com.cn/html/index.asp，遵守以下的语法规则：

```
scheme://host.domain:port/path/filename
```

解释：

- scheme - 定义因特网服务的类型。最常见的类型是 http
- host - 定义域主机（http 的默认主机是 www）
- domain - 定义因特网域名，比如 w3school.com.cn
- :port - 定义主机上的端口号（http 的默认端口号是 80）
- path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。
- filename - 定义文档/资源的名称

编者注：URL 的英文全称是 Uniform Resource Locator，中文也译为“统一资源定位符”。

URL Schemes

以下是其中一些最流行的 scheme：

Scheme	访问	用于...
http	超文本传输协议	以 http:// 开头的普通网页。不加密。
https	安全超文本传输协议	安全网页。加密所有信息交换。
ftp	文件传输协议	用于将文件下载或上传至网站。
file	您计算机上的文件。	

HTML URL 字符编码

URL 编码会将字符转换为可通过因特网传输的格式。

URL - 统一资源定位器

Web 浏览器通过 URL 从 web 服务器请求页面。

URL 是网页的地址，比如 *http://www.w3school.com.cn*。

URL 编码

URL 只能使用 ASCII 字符集来通过因特网进行发送。

由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格。

亲自试一试

如果您点击下面的“提交”按钮，浏览器会在发送输入之前对其进行 URL 编码。服务器上的页面会显示出接收到的输入。

Submit

试着输入一些字符，然后再次点击提交按钮。

URL 编码示例

字符	URL 编码
€	%80
£	%A3
?	%A9
?	%AE
à	%C0
á	%C1
?	%C2
?	%C3
?	%C4
?	%C5

如需完整的 URL 编码参考，请访问我们的 URL 编码参考手册。

HTML Web Server

如果希望向世界发布您的网站，那么您必须把它存放在 **web** 服务器上。

托管自己的网站

在自己的服务器上托管网站始终是一个选项。有几点需要考虑：

硬件支出

如果要运行“真正”的网站，您不得不购买强大的服务器硬件。不要指望低价的 PC 能够应付这些工作。您还需要稳定的（一天 24 小时）高速连接。

软件支出

请记住，服务器授权通常比客户端授权更昂贵。同时请注意，服务器授权也许有用户数量限制。

人工费

不要指望低廉的人工费用。您必须安装自己的硬件和软件。您同时要处理漏洞和病毒，以确保您的服务器时刻正常地运行于一个“任何事都可能发生”的环境中。

使用因特网服务提供商（**ISP**）

从 ISP 租用服务器也很常见。

大多数小公司会把网站存放到由 ISP 提供的服务器上。其优势有以下几点：

连接速度

大多数 ISP 都拥有连接因特网的高速连接。

强大的硬件

ISP 的 web 服务器通常强大到能够由若干网站分享资源。您还要看一下 ISP 是否提供高效的负载平衡，以及必要的备份服务器。

安全性和可靠性

ISP 是网站托管方面的专家。他们应该提供 99% 以上的在线时间，最新的软件补丁，以及最好的病毒防护。

选择 ISP 时的注意事项

24 小时支持

确保 ISP 提供 24 小时支持。不要使自己置于无法解决严重问题的尴尬境地，同时还必须等待第二个工作日。如果您不希望支付长途电话费，那么免费电话服务也是必要的。

每日备份

确保 ISP 会执行每日备份的例行工作，否则您有可能损失有价值的数据。

流量

研究一下 ISP 的流量限制。如果出现由于网站受欢迎而激增的不可预期的访问量，那么您要确保不会因此支付额外费用。

带宽或内容限制

研究一下 ISP 的带宽和内容限制。如果您计划发布图片或播出视频或音频，请确保您有此权限。

E-mail 功能

请确保 ISP 支持您需要的 e-mail 功能。

数据库访问

如果您计划使用网站数据库中的数据，那么请确保您的 ISP 支持您需要的数据库访问。

在您选取一家 ISP 之前，请务必阅读 W3School 的 Web 主机教程。

HTML 媒体

HTML 多媒体

Web 上的多媒体指的是音效、音乐、视频和动画。

现代网络浏览器已支持很多多媒体格式。

什么是多媒体？

多媒体来自多种不同的格式。它可以是您听到或看到的任何内容，文字、图片、音乐、音效、录音、电影、动画等等。

在因特网上，您会经常发现嵌入网页中的多媒体元素，现代浏览器已支持多种多媒体格式。

在本教程中，您将了解到不同的多媒体格式，以及如何在您的网页中使用它们。

浏览器支持

第一款因特网浏览器只支持文本，而且即使是对文本的支持也仅限于单一字体和单一颜色。随后诞生了支持颜色、字体和文本样式的浏览器，图片支持也被加入。

不同的浏览器以不同的方式处理对音效、动画和视频的支持。某些元素能够以内联的方式处理，而某些则需要额外的插件。

您将在下面的章节学习更多有关插件的知识。

多媒体格式

多媒体元素（比如视频和音频）存储于媒体文件中。

确定媒体类型的最常用的方法是查看文件扩展名。当浏览器得到文件扩展名 .htm 或 .html 时，它会假定该文件是 HTML 页面。.xml 扩展名指示 XML 文件，而 .css 扩展名指示样式表。图片格式则通过 .gif 或 .jpg 来识别。

多媒体元素元素也拥有带有不同扩展名的文件格式，比如 .swf、.wmv、.mp3 以及 .mp4。

视频格式

MP4 格式是一种新的即将普及的因特网视频格式。HTML5 、Flash 播放器以及优酷等视频网站均支持它。

格式	文件	描述
AVI	.avi	AVI (Audio Video Interleave) 格式是由微软开发的。所有运行 Windows 的计算机都支持 AVI 格式。它是因特网上很常见的格式，但非 Windows 计算机并不总是能够播放。
WMV	.wmv	Windows Media 格式是由微软开发的。Windows Media 在因特网上很常见，但是如果未安装额外的（免费）组件，就无法播放 Windows Media 电影。一些后期的 Windows Media 电影在所有非 Windows 计算机上都无法播放，因为没有合适的播放器。
MPEG	.mpg .mpeg	MPEG (Moving Pictures Expert Group) 格式是因特网上最流行的格式。它是跨平台的，得到了所有最流行的浏览器的支持。
QuickTime	.mov	QuickTime 格式是由苹果公司开发的。QuickTime 是因特网上常见的格式，但是 QuickTime 电影不能在没有安装额外的（免费）组件的 Windows 计算机上播放。
RealVideo	.rm .ram	RealVideo 格式是由 Real Media 针对因特网开发的。该格式允许低带宽条件下（在线视频、网络电视）的视频流。由于是低带宽优先的，质量常会降低。
Flash	.swf .flv	Flash (Shockwave) 格式是由 Macromedia 开发的。Shockwave 格式需要额外的组件来播放。但是该组件会预装到 Firefox 或 IE 之类的浏览器上。
Mpeg-4	.mp4	Mpeg-4 (with H.264 video compression) 是一种针对因特网的新格式。事实上，YouTube 推荐使用 MP4。YouTube 接收多种格式，然后全部转换为 .flv 或 .mp4 以供分发。越来越多的视频发布者转到 MP4，将其作为 Flash 播放器和 HTML5 的因特网共享格式。

声音格式

格式	文件	描述
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface) 是一种针对电子音乐设备（比如合成器和声卡）的格式。MIDI 文件不含有声音，但包含可被电子产品（比如声卡）播放的数字音乐指令。点击这里播放 The Beatles。 因为 MIDI 格式仅包含指令，所以 MIDI 文件极其小巧。上面的例子只有 23k 的大小，但却能播放将近 5 分钟。MIDI 得到了广泛的平台上的大量软件的支持。大多数流行的网络浏览器都支持 MIDI。
RealAudio	.rm .ram	RealAudio 格式是由 Real Media 针对因特网开发的。该格式也支持视频。该格式允许低带宽条件下的音频流（在线音乐、网络音乐）。由于是低带宽优先的，质量常会降低。
Wave	.wav	Wave (waveform) 格式是由 IBM 和微软开发的。所有运行 Windows 的计算机和所有网络浏览器（除了 Google Chrome）都支持它。
WMA	.wma	WMA 格式 (Windows Media Audio)，质量优于 MP3，兼容大多数播放器，除了 iPod。WMA 文件可作为连续的数据流来传输，这使它对于网络电台或在线音乐很实用。
MP3	.mp3 .mpga	MP3 文件实际上是 MPEG 文件的声音部分。MPEG 格式最初是由运动图像专家组开发的。MP3 是其中最受欢迎的针对音乐的声音格式。期待未来的软件系统都支持它。

使用哪种格式？

WAVE 是因特网上最受欢迎的无压缩声音格式，所有流行的浏览器都支持它。如果您需要未经压缩的声音（音乐或演讲），那么您应该使用 WAVE 格式。

MP3 是最新的压缩录制音乐格式。MP3 这个术语已经成为数字音乐的代名词。如果您的网址从事录制音乐，那么 MP3 是一个选项。

HTML Object 元素

<object> 的作用是支持 **HTML** 助手（插件）。

HTML 助手（插件）

辅助应用程序（helper application）是可由浏览器启动的程序。辅助应用程序也称为插件。

辅助程序可用于播放音频和视频（以及其他）。辅助程序是使用 <object> 标签来加载的。

使用辅助程序播放视频和音频的一个优势是，您能够允许用户来控制部分或全部播放设置。

大多数辅助应用程序允许对音量设置和播放功能（比如后退、暂停、停止和播放）的手工（或程序的）控制。

在 HTML 中播放视频的最好方式？

如需了解在 HTML 中包含音视频的最好方法，请参阅下一章节。

使用 QuickTime 来播放 Wave 音频

实例

```
<object width="420" height="360"
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
<param name="src" value="bird.wav" />
<param name="controller" value="true" />
</object>
```

使用 QuickTime 来播放 MP4 视频

实例

```
<object width="420" height="360"
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
<param name="src" value="movie.mp4" />
<param name="controller" value="true" />
</object>
```

使用 Flash 来播放 SWF 视频

实例

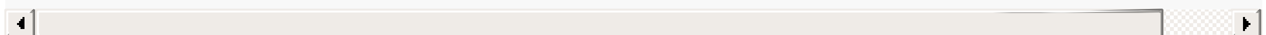
```
<object width="400" height="40"
classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/
pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">
<param name="SRC" value="bookmark.swf">
<embed src="bookmark.swf" width="400" height="40"></embed>
</object>
```

使用 Windows Media Player 来播放 WMV 影片

下面的例子展示用于播放 Windows 媒体文件的推荐代码：

实例

```
<object width="100%" height="100%"
type="video/x-ms-asf" url="3d.wmv" data="3d.wmv"
classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">
<param name="url" value="3d.wmv">
<param name="filename" value="3d.wmv">
<param name="autostart" value="1">
<param name="uiMode" value="full" />
<param name="autosize" value="1">
<param name="playcount" value="1">
<embed type="application/x-mplayer2" src="3d.wmv" width="100%"
 height="100%" autostart="true" showcontrols="true"
pluginspage="http://www.microsoft.com/Windows/MediaPlayer/"></embe
</object>
```



HTML 音频

在 **HTML** 中播放声音的方法有很多种。

问题，问题，以及解决方法

在 HTML 中播放音频并不容易！

您需要谙熟大量技巧，以确保您的音频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

使用插件

浏览器插件是一种扩展浏览器标准功能的小型计算机程序。

插件有很多用途：播放音乐、显示地图、验证银行账号，控制输入等等。

可使用 <object> 或 <embed> 标签来将插件添加到 HTML 页面。

这些标签定义资源（通常非 HTML 资源）的容器，根据类型，它们即会由浏览器显示，也会由外部插件显示。

使用 **<embed>** 元素

<embed> 标签定义外部（非 HTML）内容的容器。（这是一个 HTML5 标签，在 HTML4 中是非法的，但是所有浏览器中都有效）。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

```
<embed height="100" width="100" src="song.mp3" />
```

问题：

- <embed> 标签在 HTML 4 中是无效的。页面无法通过 HTML 4 验证。
- 不同的浏览器对音频格式的支持也不同。
- 如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。
- 如果用户的计算机未安装插件，无法播放音频。

- 如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

使用 <object> 元素

<object tag> 标签也可以定义外部（非 HTML）内容的容器。

下面的代码片段能够显示嵌入网页中的 MP3 文件：

实例

```
<object height="100" width="100" data="song.mp3"></object>
```

问题：

- 不同的浏览器对音频格式的支持也不同。
- 如果浏览器不支持该文件格式，没有插件的话就无法播放该音频。
- 如果用户的计算机未安装插件，无法播放音频。
- 如果把该文件转换为其他格式，仍然无法在所有浏览器中播放。

使用 HTML5 <audio> 元素

<audio> 元素是一个 HTML5 元素，在 HTML 4 中是非法的，但在所有浏览器中都有效。

实例

```
<audio controls="controls">
  <source src="song.mp3" type="audio/mp3" />
  <source src="song.ogg" type="audio/ogg" />
Your browser does not support this audio format.
</audio>
```

上面的例子使用了一个 mp3 文件，这样它在 Internet Explorer、Chrome 以及 Safari 中是有效的。

为了使这段音频在 Firefox 和 Opera 中同样有效，添加了一个 ogg 类型的文件。如果失败，会显示错误消息。

问题：

- <audio> 标签在 HTML 4 中是无效的。您的页面无法通过 HTML 4 验证。

- 您必须把音频文件转换为不同的格式。
- <audio> 元素在老式浏览器中不起作用。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

最好的 **HTML** 解决方法

实例

```
<audio controls="controls" height="100" width="100">
  <source src="song.mp3" type="audio/mp3" />
  <source src="song.ogg" type="audio/ogg" />
<embed height="100" width="100" src="song.mp3" />
</audio>
```

上面的例子使用了两个不同的音频格式。HTML5 <audio> 元素会尝试以 mp3 或 ogg 来播放音频。如果失败，代码将回退尝试 <embed> 元素。

问题：

- 您必须把音频转换为不同的格式。
- <audio> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法回退来显示错误消息。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

向网站添加音频的最简单方法

向网页添加音频的最简单的方法是什么？

雅虎的媒体播放器绝对算其中之一。

使用雅虎媒体播放器是一个不同的途径。您只需简单地让雅虎来完成歌曲播放的工作就好了。

它能播放 mp3 以及一系列其他格式。通过一行简单的代码，您就可以把它添加到网页中，轻松地将 HTML 页面转变为专业的播放列表。

雅虎媒体播放器

实例

```
<a href="song.mp3">Play Sound</a>

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js
</script>
```

使用雅虎播放器是免费的。如需使用它，您需要把这段 JavaScript 插入网页底部：

```
<script type="text/javascript" src="http://mediaplayer.yahoo.com/js
```

然后只需简单地把 MP3 文件链接到您的 HTML 中，JavaScript 会自动地为每首歌创建播放按钮：

```
<a href="song1.mp3">Play Song 1</a>
<a href="song2.mp3">Play Song 2</a>
...
...
...
```

雅虎媒体播放器为您的用户提供的是一个小型的播放按钮，而不是完整的播放器。不过，当您点击该按钮，会弹出完整的播放器。

请注意，这个播放器始终停靠在窗框底部。只需点击它，就可将其滑出。

使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 mp3 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”来播放该文件：

实例

```
<a href="song.mp3">Play the sound</a>
```

内联的声音

当您在网页中包含声音，或者作为网页的组成部分时，它被称为内联声音。

如果您打算在 web 应用程序中使用内联声音，您需要意识到很多人都觉得内联声音令人恼火。同时请注意，用户可能已经关闭了浏览器中的内联声音选项。

我们最好的建议是只在用户希望听到内联声音的地方包含它们。一个正面的例子是，在用户需要听到录音并点击某个链接时，会打开页面然后播放录音。

HTML 4.01 多媒体标签

标签	描述
<applet>	不赞成。定义内嵌 applet。
<embed>	HTML4 中不赞成，HTML5 中允许。定义内嵌对象。
<object>	定义内嵌对象。
<param>	定义对象的参数。

HTML 5 多媒体标签

标签	描述
<audio>	标签定义声音，比如音乐或其他音频流。
<embed>	标签定义嵌入的内容，比如插件。

HTML 视频

在 **HTML** 中播放视频的方法有很多种。

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240" />
  </object>
</video>
```

问题，问题，以及解决方法

在 HTML 中播放视频并不容易！

您需要谙熟大量技巧，以确保您的视频文件在所有浏览器中（Internet Explorer, Chrome, Firefox, Safari, Opera）和所有硬件上（PC, Mac , iPad, iPhone）都能够播放。

在本章，W3School 为您总结了问题和解决方法。

使用 **<embed>** 标签

<embed> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 代码显示嵌入网页的 Flash 视频：

实例

```
<embed src="movie.swf" height="200" width="200"/>
```

问题

- HTML4 无法识别 <embed> 标签。您的页面无法通过验证。
- 如果浏览器不支持 Flash，那么视频将无法播放
- iPad 和 iPhone 不能显示 Flash 视频。

- 如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

使用 **<object>** 标签

<object> 标签的作用是在 HTML 页面中嵌入多媒体元素。

下面的 HTML 片段显示嵌入网页的一段 Flash 视频：

实例

```
<object data="movie.swf" height="200" width="200"/>
```

问题

- 如果浏览器不支持 Flash，将无法播放视频。
- iPad 和 iPhone 不能显示 Flash 视频。
- 如果您将视频转换为其他格式，那么它仍然不能在所有浏览器中播放。

使用 **<video>** 标签

<video> 是 HTML 5 中的新标签。

<video> 标签的作用是在 HTML 页面中嵌入视频元素。

以下 HTML 片段会显示一段嵌入网页的 ogg、mp4 或 webm 格式的视频：

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
Your browser does not support the video tag.
</video>
```

问题

- 您必须把视频转换为很多不同的格式。
- <video> 元素在老式浏览器中无效。
- <video> 元素无法通过 HTML 4 和 XHTML 验证。

最好的 **HTML** 解决方法

HTML 5 + <object> + <embed>

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
  <object data="movie.mp4" width="320" height="240">
    <embed src="movie.swf" width="320" height="240" />
  </object>
</video>
```

上例中使用了 4 中不同的视频格式。HTML 5 <video> 元素会尝试播放以 mp4、ogg 或 webm 格式中的一种来播放视频。如果均失败，则回退到 <embed> 元素。

问题

- 您必须把视频转换为很多不同的格式
- <video> 元素无法通过 HTML 4 和 XHTML 验证。
- <embed> 元素无法通过 HTML 4 和 XHTML 验证。

注释：使用 <!DOCTYPE html> (HTML5) 解决验证问题。

优酷解决方案

在 HTML 中显示视频的最简单的方法是使用优酷等视频网站。

如果您希望在网页中播放视频，那么您可以把视频上传到优酷等视频网站，然后在您的网页中插入 HTML 代码即可播放视频：

```
<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v
width="480" height="400"
type="application/x-shockwave-flash">
</embed>
```

使用超链接

如果网页包含指向媒体文件的超链接，大多数浏览器会使用“辅助应用程序”来播放文件。

以下代码片段显示指向 AVI 文件的链接。如果用户点击该链接，浏览器会启动“辅助应用程序”，比如 Windows Media Player 来播放这个 AVI 文件：

实例

```
<a href="movie.swf">Play a video file</a>
```

关于内联视频的一段注释

当视频被包含在网页中时，它被称为内联视频。

如果您打算在 web 应用程序中使用内联视频，您需要意识到很多人都觉得内联视频令人恼火。

同时请注意，用户可能已经关闭了浏览器中的内联视频选项。

我们最好的建议是只在用户希望看到内联视频的地方包含它们。一个正面的例子是，在用户需要看到视频并点击某个链接时，会打开页面然后播放视频。

HTML 4.01 多媒体标签

标签	描述
<applet>	不赞成。定义内嵌 applet。
<embed>	不赞成。定义内嵌对象。（HTML5 中允许）
<object>	定义内嵌对象。
<param>	定义对象的参数。

HTML 5 多媒体标签

标签	描述
<video>	标签定义声音，比如音乐或其他音频流。
<embed>	标签定义嵌入的内容，比如插件。

HTML XHTML

XHTML 简介

XHTML 是以 XML 格式编写的 HTML。

什么是 XHTML？

- XHTML 指的是可扩展超文本标记语言
- XHTML 与 HTML 4.01 几乎是相同的
- XHTML 是更严格更纯净的 HTML 版本
- XHTML 是以 XML 应用的方式定义的 HTML
- XHTML 是 2001 年 1 月发布的 W3C 推荐标准
- XHTML 得到所有主流浏览器的支持

为什么使用 XHTML？

因特网上的很多页面包含了“糟糕”的 HTML。

如果在浏览器中查看，下面的 HTML 代码运行起来非常正常（即使它并未遵守 HTML 规则）：

```
<html>
<head>
<title>This is bad HTML</title>
<body>
<h1>Bad HTML
<p>This is a paragraph
</body>
```

XML 是一种必须正确标记且格式良好的标记语言。

如果希望学习 XML，请阅读我们的 XML 教程。

今日的科技界存在一些不同的浏览器技术。其中一些在计算机上运行，而另一些可能在移动电话或其他小型设备上运行。小型设备往往缺乏解释“糟糕”的标记语言的资源和能力。

所以 - 通过结合 XML 和 HTML 的长处，开发出了 XHTML。XHTML 是作为 XML 被重新设计的 HTML。

与 HTML 相比最重要的区别：

文档结构

- XHTML DOCTYPE 是强制性的

- <html> 中的 XML namespace 属性是强制性的
- <html>、<head>、<title> 以及 <body> 也是强制性的

元素语法

- XHTML 元素必须正确嵌套
- XHTML 元素必须始终关闭
- XHTML 元素必须小写
- XHTML 文档必须有一个根元素

属性语法

- XHTML 属性必须使用小写
- XHTML 属性值必须用引号包围
- XHTML 属性最小化也是禁止的

<!DOCTYPE ....> 是强制性的

XHTML 文档必须进行 XHTML 文档类型声明（XHTML DOCTYPE declaration）。

您可以在 W3School 的标签参考手册中找到完整的 XHTML 文档类型。

<html>、<head>、<title> 以及 <body> 元素也必须存在，并且必须使用 <html> 中的 xmlns 属性为文档规定 xml 命名空间。

下面的例子展示了带有最少的必需标签的 XHTML 文档：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Title of document</title>
</head>

<body>
......
</body>

</html>
```

如何从 HTML 转换到 XHTML

1. 向每张页面的第一行添加 XHTML <!DOCTYPE>
2. 向每张页面的 html 元素添加 xmlns 属性

3. 把所有元素名改为小写
4. 关闭所有空元素
5. 把所有属性名改为小写
6. 为所有属性值加引号

用 W3C 验证器检验 XHTML

在下面的文本框中输入您的网址：

<input name="uri" size="60" style="margin:10px 0 0 0;" value="http://www.w3school.com.cn/html/index.asp"> <input type="submit" value="验证文件">

XHTML 测验

该测验包含 20 道问题，且没有时间限制。

本测验是非正式的，它仅仅是了解您 XHTML 知识掌握程度的一个不错的途径。

每项正确答案可获得 1 分。在测试结束后，会显示您的总分。最高分为 20 分。

开始 XHTML 测验

XHTML - 元素

XHTML 元素是以 **XML** 格式编写的 **HTML** 元素。

XHTML 元素 - 语法规则

- XHTML 元素必须正确嵌套
- XHTML 元素必须始终关闭
- XHTML 元素必须小写
- XHTML 文档必须有一个根元素

XHTML 元素必须正确嵌套

在 HTML 中，某些元素可以不正确地彼此嵌套在一起，就像这样：

```
<b><i>This text is bold and italic</b></i>
```

在 XHTML 中，所有元素必须正确地彼此嵌套，就像这样：

```
<b><i>This text is bold and italic</i></b>
```

XHTML 元素必须始终关闭

这是错误的：

```
<p>This is a paragraph
<p>This is another paragraph
```

这是正确的：

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

空元素也必须关闭

这是错误的：

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

这是正确的：

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

XHTML 元素必须小写

这是错误的：

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

这是正确的：

```
<body>
<p>This is a paragraph</p>
</body>
```

XHTML - 属性

XHTML 属性是以 **XML** 格式编写的 **HTML** 属性。

XHTML 属性 - 语法规则

- XHTML 属性必须使用小写
- XHTML 属性值必须用引号包围
- XHTML 属性最小化也是禁止的

XHTML 属性必须使用小写

这是错误的：

```
<table WIDTH="100%">
```

这是正确的：

```
<table width="100%">
```

XHTML 属性值必须用引号包围

这是错误的：

```
<table width=100%>
```

这是正确的：

```
<table width="100%">
```

禁止属性简写

这是错误的：

```
<input checked>
<input readonly>
<input disabled>
<option selected>
```

这是正确的：

```
<input checked="checked" />
<input readonly="readonly" />
<input disabled="disabled" />
<option selected="selected" />
```

CSS 基础

HTML 5 简介

HTML5 是下一代的 HTML。

什么是 HTML5？

HTML5 将成为 HTML、XHTML 以及 HTML DOM 的新标准。

HTML 的上一个版本诞生于 1999 年。自从那以后，Web 世界已经经历了巨变。

HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。

HTML5 是如何起步的？

HTML5 是 W3C 与 WHATWG 合作的结果。

编者注：W3C 指 World Wide Web Consortium，万维网联盟。

编者注：WHATWG 指 Web Hypertext Application Technology Working Group。

WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

为 HTML5 建立的一些规则：

- 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- 减少对外部插件的需求（比如 Flash）
- 更优秀的错误处理
- 更多取代脚本的标记
- HTML5 应该独立于设备
- 开发进程应对公众透明

新特性

HTML5 中的一些有趣的新特性：

- 用于绘画的 canvas 元素
- 用于媒介回放的 video 和 audio 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 article、footer、header、nav、section
- 新的表单控件，比如 calendar、date、time、email、url、search

浏览器支持

最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。Internet Explorer 9 将支持某些 HTML5 特性。

HTML 5 视频

许多时髦的网站都提供视频。**HTML5** 提供了展示视频的标准。

Web 上的视频

直到现在，仍然不存在一项旨在网页上显示视频的标准。

今天，大多数视频是通过插件（比如 Flash）来显示的。然而，并非所有浏览器都拥有同样的插件。

HTML5 规定了一种通过 video 元素来包含视频的标准方法。

视频格式

当前，video 元素支持三种视频格式：

格式	IE	Firefox	Opera	Chrome	Safari
Ogg	No	3.5+	10.5+	5.0+	No
MPEG 4	9.0+	No	No	5.0+	3.0+
WebM	No	4.0+	10.6+	6.0+	No

Ogg = 带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件

MPEG4 = 带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件

WebM = 带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件

如何工作

如需在 HTML5 中显示视频，您所有需要的是：

```
<video src="movie.ogg" controls="controls">
</video>
```

control 属性供添加播放、暂停和音量控件。

包含宽度和高度属性也是不错的主意。

<video> 与 </video> 之间插入的内容是供不支持 video 元素的浏览器显示的：

实例

```
<video src="movie.ogg" width="320" height="240" controls="controls'
Your browser does not support the video tag.
</video>
```

上面的例子使用一个 Ogg 文件，适用于Firefox、Opera 以及 Chrome 浏览器。

要确保适用于 Safari 浏览器，视频文件必须是 MPEG4 类型。

video 元素允许多个 source 元素。source 元素可以链接不同的视频文件。浏览器将使用第一个可识别的格式：

实例

```
<video width="320" height="240" controls="controls">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

Internet Explorer

Internet Explorer 8 不支持 video 元素。在 IE 9 中，将提供对使用 MPEG4 的 video 元素的支持。

<video> 标签的属性

属性	值	描述
autoplay	autoplay	如果出现该属性，则视频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
height	*pixels*	设置视频播放器的高度。
loop	loop	如果出现该属性，则当媒介文件完成播放后再次开始播放。
preload	preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放。 如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的视频的 URL。
width	*pixels*	设置视频播放器的宽度。

相关页面

参考手册：HTML 5 <video> 标签

HTML 5 Video + DOM

HTML5 <video> - 使用 DOM 进行控制

HTML5 <video> 元素同样拥有方法、属性和事件。

其中的方法用于播放、暂停以及加载等。其中的属性（比如时长、音量等）可以被读取或设置。其中的 DOM 事件能够通知您，比方说，<video> 元素开始播放、已暂停，已停止，等等。

下例中简单的方法，向我们演示了如何使用 <video> 元素，读取并设置属性，以及如何调用方法。

实例

为视频创建简单的播放/暂停以及调整尺寸控件：

```
<!DOCTYPE html>
<html>
<body>

<div style="text-align:center;">
  <button onclick="playPause()">播放/暂停</button>
  <button onclick="makeBig()">大</button>
  <button onclick="makeNormal()">中</button>
  <button onclick="makeSmall()">小</button>
  <br />
  <video id="video1" width="420" style="margin-top:15px;">
    <source src="/example/html5/mov_bbb.mp4" type="video/mp4" />
    <source src="/example/html5/mov_bbb.ogg" type="video/ogg" />
    Your browser does not support HTML5 video.
  </video>
</div>

<script type="text/javascript">
var myVideo=document.getElementById("video1");

function playPause()
{
if (myVideo.paused)
  myVideo.play();
else
  myVideo.pause();
}

function makeBig()
{
myVideo.width=560;
}

function makeSmall()
{
myVideo.width=320;
}

function makeNormal()
{
myVideo.width=420;
}
</script>

</body>
</html>
```

上面的例子调用了两个方法：play() 和 pause()。它同时使用了两个属性：paused 和 width。

HTML5 <video> - 方法、属性以及事件

下面列出了大多数浏览器支持的视频方法、属性和事件：

方法	属性	事件
play()	currentSrc	play
pause()	currentTime	pause
load()	videoWidth	progress
canPlayType	videoHeight	error
duration	timeupdate	
ended	ended	
error	abort	
paused	empty	
muted	emptied	
seeking	waiting	
volume	loadedmetadata	
height		
width		

注释：在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

HTML 5 音频

HTML5 提供了播放音频的标准。

Web 上的音频

直到现在，仍然不存在一项旨在网页上播放音频的标准。

今天，大多数音频是通过插件（比如 Flash）来播放的。然而，并非所有浏览器都拥有同样的插件。

HTML5 规定了一种通过 audio 元素来包含音频的标准方法。

audio 元素能够播放声音文件或者音频流。

音频格式

当前，audio 元素支持三种音频格式：

IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis	√	√	√	
MP3	√	√	√	
Wav	√	√	√	

如何工作

如需在 HTML5 中播放音频，您所有需要的是：

```
<audio src="song.ogg" controls="controls">
</audio>
```

control 属性供添加播放、暂停和音量控件。

<audio> 与 </audio> 之间插入的内容是供不支持 audio 元素的浏览器显示的：

实例

```
<audio src="song.ogg" controls="controls">
Your browser does not support the audio tag.
</audio>
```

上面的例子使用一个 Ogg 文件，适用于Firefox、Opera 以及 Chrome 浏览器。

要确保适用于 Safari 浏览器，音频文件必须是 MP3 或 Wav 类型。

audio 元素允许多个 source 元素。source 元素可以链接不同的音频文件。浏览器将使用第一个可识别的格式：

实例

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg">
  <source src="song.mp3" type="audio/mpeg">
Your browser does not support the audio tag.
</audio>
```

Internet Explorer

Internet Explorer 8 不支持 audio 元素。在 IE 9 中，将提供对 audio 元素的支持。

<audio> 标签的属性

属性	值	描述
autoplay	autoplay	如果出现该属性，则音频在就绪后马上播放。
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮。
loop	loop	如果出现该属性，则每当音频结束时重新开始播放。
preload	preload	如果出现该属性，则音频在页面加载时进行加载，并预备播放。 如果使用 "autoplay"，则忽略该属性。
src	*url*	要播放的音频的 URL。

相关页面

参考手册：HTML 5 <audio> 标签

HTML 5 拖放

拖放（**Drag** 和 **drop**）是 **HTML5** 标准的组成部分。

拖放

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。

在 HTML5 中，拖放是标准的一部分，任何元素都能够拖放。

浏览器支持

Internet Explorer 9、Firefox、Opera 12、Chrome 以及 Safari 5 支持拖放。

注释：在 Safari 5.1.2 中不支持拖放。

HTML5 拖放实例

下面的例子是一个简单的拖放实例：

实例

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}

function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

它看上去也许有些复杂，不过我们可以分别研究拖放事件的不同部分。

设置元素为可拖放

首先，为了使元素可拖动，把 draggable 属性设置为 true ：

```
<img draggable="true" />
```

拖动什么 - ondragstart 和 setData()

然后，规定当元素被拖动时，会发生什么。

在上面的例子中，ondragstart 属性调用了一个函数，drag(*event*)，它规定了被拖动的数据。

dataTransfer.setData() 方法设置被拖数据的数据类型和值：

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

在这个例子中，数据类型是 "Text"，值是可拖动元素的 id ("drag1")。

放到何处 - ondragover

ondragover 事件规定在何处放置被拖动的数据。

默认地，无法将数据/元素放置到其他元素中。如果需要设置允许放置，我们必须阻止对元素的默认处理方式。

这要通过调用 ondragover 事件的 *event*.preventDefault() 方法：

```
event.preventDefault()
```

进行放置 - ondrop

当放置被拖数据时，会发生 drop 事件。

在上面的例子中，ondrop 属性调用了一个函数，drop(*event*)：

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
```

代码解释：

- 调用 preventDefault() 来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）
- 通过 dataTransfer.getData("Text") 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据。
- 被拖数据是被拖元素的 id ("drag1")
- 把被拖元素追加到放置元素（目标元素）中

更多实例

来回拖放图片

HTML 5 Canvas

canvas 元素用于在网页上绘制图形。

什么是 Canvas？

HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。

画布是一个矩形区域，您可以控制其每一像素。

canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

创建 Canvas 元素

向 HTML5 页面添加 canvas 元素。

规定元素的 id、宽度和高度：

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

通过 JavaScript 来绘制

canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成：

```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
</script>
```

JavaScript 使用 id 来寻找 canvas 元素：

```
var c=document.getElementById("myCanvas");
```

然后，创建 context 对象：

```
var cxt=c.getContext("2d");
```

getContext("2d") 对象是内建的 HTML5 对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

下面的两行代码绘制一个红色的矩形：

```
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
```

fillStyle 方法将其染成红色，fillRect 方法规定了形状、位置和尺寸。

理解坐标

上面的 fillRect 方法拥有参数 (0,0,150,75)。

意思是：在画布上绘制 150x75 的矩形，从左上角开始 (0,0)。

如下图所示，画布的 X 和 Y 坐标用于在画布上对绘画进行定位。



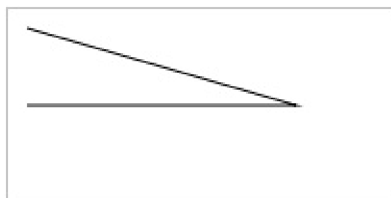
实例：把鼠标悬停在矩形上可以看到坐标

更多 Canvas 实例

下面的在 canvas 元素上进行绘画的更多实例：

实例 - 线条

通过指定从何处开始，在何处结束，来绘制一条线：



JavaScript 代码：

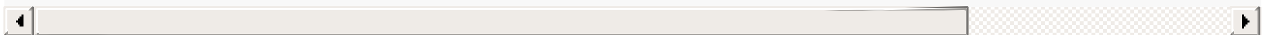
```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.moveTo(10,10);
cxt.lineTo(150,50);
cxt.lineTo(10,50);
cxt.stroke();

</script>
```

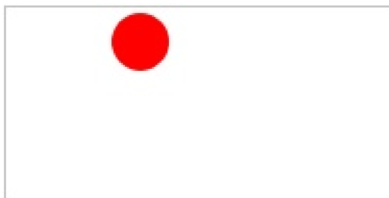
canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px sc
Your browser does not support the canvas element.
</canvas>
```



实例 - 圆形

通过规定尺寸、颜色和位置，来绘制一个圆：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#FF0000";
cxt.beginPath();
cxt.arc(70,18,15,0,Math.PI*2,true);
cxt.closePath();
cxt.fill();

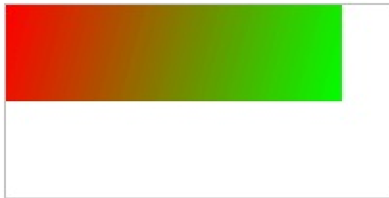
</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px sc
Your browser does not support the canvas element.
</canvas>
```

实例 - 渐变

使用您指定的颜色来绘制渐变背景：



JavaScript 代码：

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var grd=cxt.createLinearGradient(0,0,175,50);
grd.addColorStop(0,"#FF0000");
grd.addColorStop(1,"#00FF00");
cxt.fillStyle=grd;
cxt.fillRect(0,0,175,50);

</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px sc
Your browser does not support the canvas element.
</canvas>
```

实例 - 图像

把一幅图像放置到画布上：



JavaScript 代码：

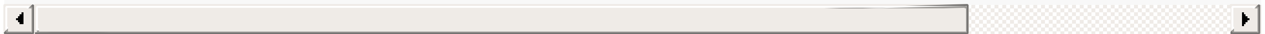
```
<script type="text/javascript">

var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var img=new Image()
img.src="flower.png"
cxt.drawImage(img,0,0);

</script>
```

canvas 元素：

```
<canvas id="myCanvas" width="200" height="100" style="border:1px so
Your browser does not support the canvas element.
</canvas>
```



相关页面

参考手册：HTML 5 <canvas> 标签

参考手册：HTML DOM Canvas 对象

HTML5 内联 SVG

HTML5 支持内联 SVG。

什么是SVG？

- SVG 指可伸缩矢量图形 (Scalable Vector Graphics)
- SVG 用于定义用于网络的基于矢量的图形
- SVG 使用 XML 格式定义图形
- SVG 图像在放大或改变尺寸的情况下其图形质量不会有损失
- SVG 是万维网联盟的标准

SVG 的优势

与其他图像格式相比（比如 JPEG 和 GIF），使用 SVG 的优势在于：

- SVG 图像可通过文本编辑器来创建和修改
- SVG 图像可被搜索、索引、脚本化或压缩
- SVG 是可伸缩的
- SVG 图像可在任何的分辨率下被高质量地打印
- SVG 可在图像质量不下降的情况下被放大

浏览器支持

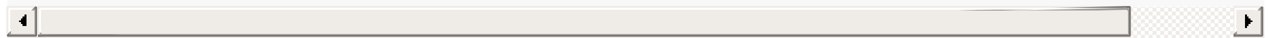
Internet Explorer 9、Firefox、Opera、Chrome 以及 Safari 支持内联 SVG。

把 SVG 直接嵌入 HTML 页面

在 HTML5 中，您能够将 SVG 元素直接嵌入 HTML 页面中：

实例

```
<!DOCTYPE html>
<html>
<body>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190'
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;'
</svg>
</body>
</html>
```



结果：

如需学习更多有关 SVG 的知识，请阅读我们的 SVG 教程。

HTML 5 Canvas vs. SVG

Canvas 和 **SVG** 都允许您在浏览器中创建图形，但是它们在根本上是不同的。

SVG

SVG 是一种使用 XML 描述 2D 图形的语言。

SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

Canvas

Canvas 通过 JavaScript 来绘制 2D 图形。

Canvas 是逐像素进行渲染的。

在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

Canvas 与 SVG 的比较

下表列出了 canvas 与 SVG 之间的一些不同之处。

Canvas

- 依赖分辨率
- 不支持事件处理器
- 弱的文本渲染能力
- 能够以 .png 或 .jpg 格式保存结果图像
- 最适合图像密集型的游戏，其中的许多对象会被频繁重绘

SVG

- 不依赖分辨率
- 支持事件处理器
- 最适合带有大型渲染区域的应用程序（比如谷歌地图）
- 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
- 不适合游戏应用

HTML5 地理定位

HTML5 Geolocation（地理定位）用于定位用户的位置。

亲自试一试：在谷歌地图上显示您的位置

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">点击这个按钮，获得您的位置：</p>
<button onclick="getLocation()">试一下</button>
<div id="mapholder"></div>
<script src="http://maps.google.com/maps/api/js?sensor=false"></sci
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition,showErroi
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }

function showPosition(position)
  {
  lat=position.coords.latitude;
  lon=position.coords.longitude;
  latlon=new google.maps.LatLng(lat, lon)
  mapholder=document.getElementById('mapholder')
  mapholder.style.height='250px';
  mapholder.style.width='500px';

  var myOptions={
  center:latlon,zoom:14,
  mapTypeId:google.maps.MapTypeId.ROADMAP,
  mapTypeControl:false,
  navigationControlOptions:{style:google.maps.NavigationControlSty]
  };
  var map=new google.maps.Map(document.getElementById("mapholder"),
  var marker=new google.maps.Marker({position:latlon,map:map,title
  }

function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="User denied the request for Geolocation."
```

```
        break;
      case error.POSITION_UNAVAILABLE:
        x.innerHTML="Location information is unavailable."
        break;
      case error.TIMEOUT:
        x.innerHTML="The request to get user location timed out."
        break;
      case error.UNKNOWN_ERROR:
        x.innerHTML="An unknown error occurred."
        break;
      }
    }
</script>
</body>
</html>
```

定位用户的位置

HTML5 Geolocation API 用于获得用户的地理位置。

鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

浏览器支持

Internet Explorer 9、Firefox、Chrome、Safari 以及 Opera 支持地理定位。

注释：对于拥有 GPS 的设备，比如 iPhone，地理定位更加精确。

HTML5 - 使用地理定位

请使用 getCurrentPosition() 方法来获得用户的位置。

下例是一个简单的地理定位实例，可返回用户位置的经度和纬度。

实例

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.getCurrentPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br />Longitude: " + position.coords.longitude;
  }
</script>
```

例子解释：

- 检测是否支持地理定位
- 如果支持，则运行 getCurrentPosition() 方法。如果不支持，则向用户显示一段消息。
- 如果getCurrentPosition()运行成功，则向参数showPosition中规定的函数返回一个coordinates对象
- showPosition() 函数获得并显示经度和纬度

上面的例子是一个非常基础的地理定位脚本，不含错误处理。

处理错误和拒绝

getCurrentPosition() 方法的第二个参数用于处理错误。它规定当获取用户位置失败时运行的函数：

实例

```
function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      x.innerHTML="User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML="Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML="The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML="An unknown error occurred."
      break;
    }
  }
```

错误代码：

- Permission denied - 用户不允许地理定位
- Position unavailable - 无法获取当前位置
- Timeout - 操作超时

在地图中显示结果

如需在地图中显示结果，您需要访问可使用经纬度的地图服务，比如谷歌地图或百度地图：

实例

```
function showPosition(position)
{
var latlon=position.coords.latitude+","+position.coords.longitude;

var img_url="http://maps.googleapis.com/maps/api/staticmap?center='
+latlon+"&zoom=14&size=400x300&sensor=false";

document.getElementById("mapholder").innerHTML="<img src='"+img_ur
}
```

在上例中，我们使用返回的经纬度数据在谷歌地图中显示位置（使用静态图像）。

谷歌地图脚本

上面的链接向您演示如何使用脚本来显示带有标记、缩放和拖曳选项的交互式地图。

给定位置的信息

本页演示的是如何在地图上显示用户的位置。不过，地理定位对于给定位置的信息同样很有用处。

案例：

- 更新本地信息
- 显示用户周围的兴趣点
- 交互式车载导航系统 (GPS)

getCurrentPosition() 方法 - 返回数据

若成功，则 getCurrentPosition() 方法返回对象。始终会返回 latitude、longitude 以及 accuracy 属性。如果可用，则会返回其他下面的属性。

属性	描述
coords.latitude	十进制数的纬度
coords.longitude	十进制数的经度
coords.accuracy	位置精度
coords.altitude	海拔，海平面以上以米计
coords.altitudeAccuracy	位置的海拔精度
coords.heading	方向，从正北开始以度计
coords.speed	速度，以米/每秒计
timestamp	响应的日期/时间

Geolocation 对象 - 其他有趣的方法

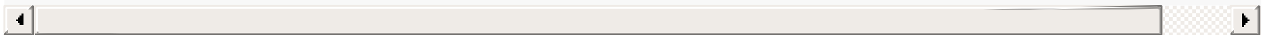
watchPosition() - 返回用户的当前位置，并继续返回用户移动时的更新位置（就像汽车上的 GPS）。

clearWatch() - 停止 watchPosition() 方法

下面的例子展示 watchPosition() 方法。您需要一台精确的 GPS 设备来测试该例（比如 iPhone）：

实例

```
<script>
var x=document.getElementById("demo");
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.watchPosition(showPosition);
    }
  else{x.innerHTML="Geolocation is not supported by this browser.";
  }
function showPosition(position)
  {
  x.innerHTML="Latitude: " + position.coords.latitude +
  "<br />Longitude: " + position.coords.longitude;
  }
</script>
```



HTML 5 Web 存储

在客户端存储数据

HTML5 提供了两种在客户端存储数据的新方法：

- localStorage - 没有时间限制的数据存储
- sessionStorage - 针对一个 session 的数据存储

之前，这些都是由 cookie 完成的。但是 cookie 不适合大量数据的存储，因为它们由每个对服务器的请求来传递，这使得 cookie 速度很慢而且效率也不高。

在 HTML5 中，数据不是由每个服务器请求传递的，而是只有在请求时使用数据。它使在不影响网站性能的情况下存储大量数据成为可能。

对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

HTML5 使用 JavaScript 来存储和访问数据。

localStorage 方法

localStorage 方法存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用。

如何创建和访问 localStorage：

实例

```
<script type="text/javascript">
localStorage.lastname="Smith";
document.write(localStorage.lastname);
</script>
```

下面的例子对用户访问页面的次数进行计数：

实例

```
<script type="text/javascript">
if (localStorage.pagecount)
  {
  localStorage.pagecount=Number(localStorage.pagecount) +1;
  }
else
  {
  localStorage.pagecount=1;
  }
document.write("Visits "+ localStorage.pagecount + " time(s).");
</script>
```

sessionStorage 方法

sessionStorage 方法针对一个 session 进行数据存储。当用户关闭浏览器窗口后，数据会被删除。

如何创建并访问一个 sessionStorage：

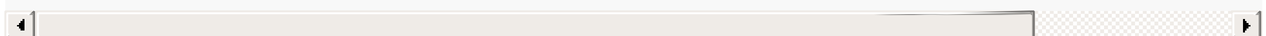
实例

```
<script type="text/javascript">
sessionStorage.lastname="Smith";
document.write(sessionStorage.lastname);
</script>
```

下面的例子对用户在当前 session 中访问页面的次数进行计数：

实例

```
<script type="text/javascript">
if (sessionStorage.pagecount)
  {
  sessionStorage.pagecount=Number(sessionStorage.pagecount) +1;
  }
else
  {
  sessionStorage.pagecount=1;
  }
document.write("Visits "+sessionStorage.pagecount+" time(s) this se
</script>
```



HTML 5 应用程序缓存

使用 **HTML5**，通过创建 **cache manifest** 文件，可以轻松地创建 **web** 应用的离线版本。

什么是应用程序缓存（Application Cache）？

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。

应用程序缓存为应用带来三个优势：

- 离线浏览 - 用户可在应用离线时使用它们
- 速度 - 已缓存资源加载得更快
- 减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

浏览器支持

所有主流浏览器均支持应用程序缓存，除了 Internet Explorer。

HTML5 Cache Manifest 实例

下面的例子展示了带有 cache manifest 的 HTML 文档（供离线浏览）：

实例

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">

<body>
The content of the document......
</body>

</html>
```

Cache Manifest 基础

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

manifest 文件的建议的文件扩展名是：".appcache"。

请注意，manifest 文件需要配置正确的 *MIME-type*，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

- *CACHE MANIFEST* - 在此标题下列出的文件将在首次下载后进行缓存
- *NETWORK* - 在此标题下列出的文件需要与服务器的连接，且不会被缓存
- *FALLBACK* - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

CACHE MANIFEST

第一行，CACHE MANIFEST，是必需的：

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

NETWORK

下面的 NETWORK 小节规定文件 "login.asp" 永远不会被缓存，且离线时是不可用的：

```
NETWORK:
login.asp
```

可以使用星号来指示所有其他资源/文件都需要因特网连接：

```
NETWORK:
*
```

FALLBACK

下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 "offline.html" 替代 /html5/ 目录中的所有文件：

```
FALLBACK:
/html5/ /404.html
```

注释：第一个 URI 是资源，第二个是替补。

更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

- 用户清空浏览器缓存
- manifest 文件被修改（参阅下面的提示）
- 由程序来更新应用缓存

实例 - 完整的 Manifest 文件

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html5/ /404.html
```

重要的提示：以 "#" 开头的是注释行，但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器

重新缓存文件的办法。

关于应用程序缓存的注释

请留心缓存的内容。

一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使您修改了服务器上的文件。为了确保浏览器更新缓存，您需要更新 manifest 文件。

注释：浏览器对缓存数据的容量限制可能不太一样（某些浏览器设置的限制是每个站点 5MB）。

HTML 5 Web Workers

web worker 是运行在后台的 **JavaScript**，不会影响页面的性能。

什么是 **Web Worker**？

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成。

web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行。

浏览器支持

所有主流浏览器均支持 web worker，除了 Internet Explorer。

HTML5 Web Workers 实例

下面的例子创建了一个简单的 web worker，在后台计数：

计数：

<button onclick="startWorker()" style="font:12px Verdana, Arial, Helvetica, sans-serif;">启动 Worker</button> <button onclick="stopWorker()" style="font:12px Verdana, Arial, Helvetica, sans-serif;">停止 Worker</button>

检测 **Web Worker** 支持

在创建 web worker 之前，请检测用户的浏览器是否支持它：

```
if(typeof(Worker)!=="undefined")
  {
  // Yes! Web worker support!
  // Some code.....
  }
else
  {
  // Sorry! No Web Worker support..
  }
```

创建 **web worker** 文件

现在，让我们在一个外部 JavaScript 中创建我们的 web worker。

在这里，我们创建了计数脚本。该脚本存储于 "demo_workers.js" 文件中：

```
var i=0;

function timedCount()
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",500);
}

timedCount();
```

以上代码中重要的部分是 *postMessage()* 方法 - 它用于向 HTML 页面传回一段消息。

注释：web worker 通常不用于如此简单的脚本，而是用于更耗费 CPU 资源的任务。

创建 Web Worker 对象

我们已经有了 web worker 文件，现在我们需要从 HTML 页面调用它。

下面的代码检测是否存在 worker，如果不存在，- 它会创建一个新的 web worker 对象，然后运行 "demo_workers.js" 中的代码：

```
if(typeof(w)=="undefined")
  {
  w=new Worker("demo_workers.js");
  }
```

然后我们就可以从 web worker 发生和接收消息了。

向 web worker 添加一个 "onmessage" 事件监听器：

```
w.onmessage=function(event){
document.getElementById("result").innerHTML=event.data;
};
```

当 web worker 传递消息时，会执行事件监听器中的代码。event.data 中存有来自 event.data 的数据。

终止 Web Worker

当我们创建 web worker 对象后，它会继续监听消息（即使在外部脚本完成之后）直到其被终止为止。

如需终止 web worker，并释放浏览器/计算机资源，请使用 terminate() 方法：

```
w.terminate();
```

完整的 Web Worker 实例代码

我们已经看到了 .js 文件中的 Worker 代码。下面是 HTML 页面的代码：

实例

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<br /><br />

<script>
var w;

function startWorker()
{
if(typeof(Worker)!=="undefined")
{
  if(typeof(w)=="undefined")
    {
    w=new Worker("demo_workers.js");
    }
  w.onmessage = function (event) {
    document.getElementById("result").innerHTML=event.data;
  };
}
else
{
document.getElementById("result").innerHTML="Sorry, your browser
 does not support Web Workers...";
}
}

function stopWorker()
{
w.terminate();
}
</script>

</body>
</html>
```

Web Workers 和 DOM

由于 web worker 位于外部文件中，它们无法访问下例 JavaScript 对象：

- window 对象
- document 对象
- parent 对象

HTML 5 服务器发送事件

HTML5 服务器发送事件（**server-sent event**）允许网页获得来自服务器的更新。

Server-Sent 事件 - 单向消息传递

Server-Sent 事件指的是网页自动获取来自服务器的更新。

以前也可能做到这一点，前提是网页不得不询问是否有可用的更新。通过服务器发送事件，更新能够自动到达。

例子：Facebook/Twitter 更新、估价更新、新的博文、赛事结果等。

浏览器支持

所有主流浏览器均支持服务器发送事件，除了 Internet Explorer。

接收 Server-Sent 事件通知

EventSource 对象用于接收服务器发送事件通知：

实例

```
var source=new EventSource("demo_sse.php");
source.onmessage=function(event)
  {
  document.getElementById("result").innerHTML+=event.data + "<br />
  };
```

例子解释：

- 创建一个新的 EventSource 对象，然后规定发送更新的页面的 URL（本例中是 "demo_sse.php"）
- 每接收到一次更新，就会发生 onmessage 事件
- 当 onmessage 事件发生时，把已接收的数据推入 id 为 "result" 的元素中

检测 Server-Sent 事件支持

在上面的 TIY 实例中，我们编写了一段额外的代码来检测服务器发送事件的浏览器支持情况：

```
if(typeof(EventSource)!=="undefined")
  {
  // Yes! Server-sent events support!
  // Some code.....
  }
else
  {
  // Sorry! No server-sent events support..
  }
```

服务器端代码实例

为了让上面的例子可以运行，您还需要能够发送数据更新的服务器（比如 PHP 和 ASP）。

服务器端事件流的语法是非常简单的。把 "Content-Type" 报头设置为 "text/event-stream"。现在，您可以开始发送事件流了。

PHP 代码 (demo_sse.php)：

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

ASP 代码 (VB) (demo_sse.asp):

```
<%
Response.ContentType="text/event-stream"
Response.Expires=-1
Response.Write("data: " & now())
Response.Flush()
%>
```

代码解释：

- 把报头 "Content-Type" 设置为 "text/event-stream"
- 规定不对页面进行缓存
- 输出发送日期（始终以 "data: " 开头）
- 向网页刷新输出数据

EventSource 对象

在上面的例子中，我们使用 onmessage 事件来获取消息。不过还可以使用其他事件：

事件	描述
onopen	当通往服务器的连接被打开
onmessage	当接收到消息
onerror	当错误发生

HTML5 Input 类型

HTML5 新的 Input 类型

HTML5 拥有多个新的表单输入类型。这些新特性提供了更好的输入控制和验证。

本章全面介绍这些新的输入类型：

- email
- url
- number
- range
- Date pickers (date, month, week, time, datetime, datetime-local)
- search
- color

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
email	No	4.0	9.0	10.0	No
url	No	4.0	9.0	10.0	No
number	No	No	9.0	7.0	No
range	No	No	9.0	4.0	4.0
Date pickers	No	No	9.0	10.0	No
search	No	4.0	11.0	10.0	No
color	No	No	11.0	No	No

注释：Opera 对新的输入类型的支持最好。不过您已经可以在所有主流的浏览器中使用它们了。即使不被支持，仍然可以显示为常规的文本域。

Input 类型 - email

email 类型用于应该包含 e-mail 地址的输入域。

在提交表单时，会自动验证 email 域的值。

实例

```
E-mail: <input type="email" name="user_email" />
```

提示：iPhone 中的 Safari 浏览器支持 email 输入类型，并通过改变触摸屏键盘来配合它（添加 @ 和 .com 选项）。

Input 类型 - url

url 类型用于应该包含 URL 地址的输入域。

在提交表单时，会自动验证 url 域的值。

实例

```
Homepage: <input type="url" name="user_url" />
```

提示：iPhone 中的 Safari 浏览器支持 url 输入类型，并通过改变触摸屏键盘来配合它（添加 .com 选项）。

Input 类型 - number

number 类型用于应该包含数值的输入域。

您还能够设定对所接受的数字的限定：

实例

```
Points: <input type="number" name="points" min="1" max="10" />
```

请使用下面的属性来规定对数字类型的限定：

属性	值	描述
max	*number*	规定允许的最大值
min	*number*	规定允许的最小值
step	*number*	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	*number*	规定默认值

请试一下带有所有限定属性的例子：

提示：iPhone 中的 Safari 浏览器支持 number 输入类型，并通过改变触摸屏键盘来配合它（显示数字）。

Input 类型 - range

range 类型用于应该包含一定范围内数字值的输入域。

range 类型显示为滑动条。

您还能够设定对所接受的数字的限定：

实例

```
<input type="range" name="points" min="1" max="10" />
```

请使用下面的属性来规定对数字类型的限定：

属性	值	描述
max	*number*	规定允许的最大值
min	*number*	规定允许的最小值
step	*number*	规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value	*number*	规定默认值

Input 类型 - Date Pickers（日期选择器）

HTML5 拥有多个可供选取日期和时间的新输入类型：

- date - 选取日、月、年
- month - 选取月、年
- week - 选取周和年
- time - 选取时间（小时和分钟）
- datetime - 选取时间、日、月、年（UTC 时间）
- datetime-local - 选取时间、日、月、年（本地时间）

下面的例子允许您从日历中选取一个日期：

实例

```
Date: <input type="date" name="user_date" />
```

输入类型 "month":

输入类型 "week":

输入类型 "time":

输入类型 "datetime":

输入类型 "datetime-local":

Input 类型 - search

search 类型用于搜索域，比如站点搜索或 Google 搜索。

search 域显示为常规的文本域。

HTML5 表单元素

HTML5 的新的表单元素：

HTML5 拥有若干涉及表单的元素和属性。

本章介绍以下新的表单元素：

- datalist
- keygen
- output

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
datalist	No	No	9.5	No	No
keygen	No	No	10.5	3.0	No
output	No	No	9.5	No	No

datalist 元素

datalist 元素规定输入域的选项列表。

列表是通过 datalist 内的 option 元素创建的。

如需把 datalist 绑定到输入域，请用输入域的 list 属性引用 datalist 的 id：

实例

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3School" value="http://www.W3School.com.cn" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

提示：option 元素永远都要设置 value 属性。

keygen 元素

keygen 元素的作用是提供一种验证用户的可靠方法。

keygen 元素是密钥对生成器（key-pair generator）。当提交表单时，会生成两个键，一个是私钥，一个公钥。

私钥（private key）存储于客户端，公钥（public key）则被发送到服务器。公钥可用于之后验证用户的客户端证书（client certificate）。

目前，浏览器对此元素的糟糕的支持度不足以使其成为一种有用的安全标准。

实例

```
<form action="demo_form.asp" method="get">
Username: <input type="text" name="usr_name" />
Encryption: <keygen name="security" />
<input type="submit" />
</form>
```

output 元素

output 元素用于不同类型的输出，比如计算或脚本输出：

实例

```
<output id="result" onforminput="resCalc()"></output>
```

HTML5 表单属性

HTML5 的新的表单属性

本章讲解涉及 <form> 和 <input> 元素的新属性。

新的 **form** 属性：

- autocomplete
- novalidate

新的 **input** 属性：

- autocomplete
- autofocus
- form
- form overrides (formaction, formenctype, formmethod, formnovalidate, formtarget)
- height 和 width
- list
- min, max 和 step
- multiple
- pattern (regexp)
- placeholder
- required

浏览器支持

Input type	IE	Firefox	Opera	Chrome	Safari
autocomplete	8.0	3.5	9.5	3.0	4.0
autofocus	No	No	10.0	3.0	4.0
form	No	No	9.5	No	No
form overrides	No	No	10.5	No	No
height and width	8.0	3.5	9.5	3.0	4.0
list	No	No	9.5	No	No
min, max and step	No	No	9.5	3.0	No
multiple	No	3.5	No	3.0	4.0
novalidate	No	No	No	No	No
pattern	No	No	9.5	3.0	No
placeholder	No	No	No	3.0	3.0
required	No	No	9.5	3.0	No

autocomplete 属性

autocomplete 属性规定 form 或 input 域应该拥有自动完成功能。

注释：autocomplete 适用于 <form> 标签，以及以下类型的 <input> 标签：text, search, url, telephone, email, password, datepickers, range 以及 color。

当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项：

实例

```
<form action="demo_form.asp" method="get" autocomplete="on">
First name: <input type="text" name="fname" /><br />
Last name: <input type="text" name="lname" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br /
<input type="submit" />
</form>
```

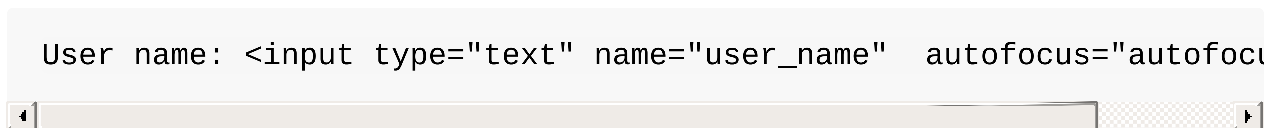
注释：在某些浏览器中，您可能需要启用自动完成功能，以使该属性生效。

autofocus 属性

autofocus 属性规定在页面加载时，域自动地获得焦点。

注释：autofocus 属性适用于所有 <input> 标签的类型。

实例



```
User name: <input type="text" name="user_name"  autofocus="autofocu
```

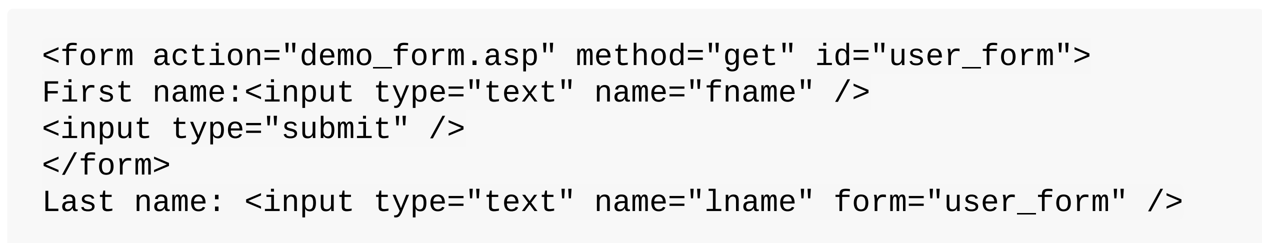
form 属性

form 属性规定输入域所属的一个或多个表单。

注释：form 属性适用于所有 <input> 标签的类型。

form 属性必须引用所属表单的 id：

实例



```
<form action="demo_form.asp" method="get" id="user_form">
First name:<input type="text" name="fname" />
<input type="submit" />
</form>
Last name: <input type="text" name="lname" form="user_form" />
```

注释：如需引用一个以上的表单，请使用空格分隔的列表。

表单重写属性

表单重写属性（form override attributes）允许您重写 form 元素的某些属性设定。

表单重写属性有：

- formaction - 重写表单的 action 属性
- formenctype - 重写表单的 enctype 属性
- formmethod - 重写表单的 method 属性
- formnovalidate - 重写表单的 novalidate 属性
- formtarget - 重写表单的 target 属性

注释：表单重写属性适用于以下类型的 <input> 标签：submit 和 image。

实例

```
<form action="demo_form.asp" method="get" id="user_form">
E-mail: <input type="email" name="userid" /><br />
<input type="submit" value="Submit" />
<br />
<input type="submit" formaction="demo_admin.asp" value="Submit as a
<br />
<input type="submit" formnovalidate="true" value="Submit without va
<br />
</form>
```

注释：这些属性对于创建不同的提交按钮很有帮助。

height 和 width 属性

height 和 width 属性规定用于 image 类型的 input 标签的图像高度和宽度。

注释：height 和 width 属性只适用于 image 类型的 <input> 标签。

实例

```
<input type="image" src="img_submit.gif" width="99" height="99" />
```

list 属性

list 属性规定输入域的 datalist。datalist 是输入域的选项列表。

注释：list 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, date pickers, number, range 以及 color。

实例

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3Schools" value="http://www.w3school.com.cn" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

min、max 和 step 属性

min、max 和 step 属性用于为包含数字或日期的 input 类型规定限定（约束）。

max 属性规定输入域所允许的最大值。

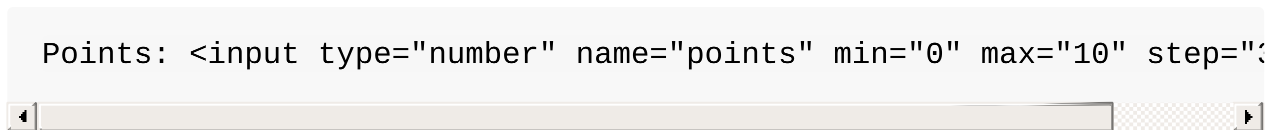
min 属性规定输入域所允许的最小值。

step 属性为输入域规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）。

注释：min、max 和 step 属性适用于以下类型的 <input> 标签：date pickers、number 以及 range。

下面的例子显示一个数字域，该域接受介于 0 到 10 之间的值，且步进为 3（即合法的值为 0、3、6 和 9）：

实例



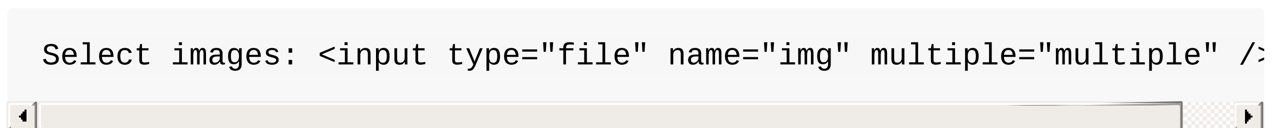
```
Points: <input type="number" name="points" min="0" max="10" step="3
```

multiple 属性

multiple 属性规定输入域中可选择多个值。

注释：multiple 属性适用于以下类型的 <input> 标签：email 和 file。

实例



```
Select images: <input type="file" name="img" multiple="multiple" />
```

novalidate 属性

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。

注释：novalidate 属性适用于 <form> 以及以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, range 以及 color.

实例

```
<form action="demo_form.asp" method="get" novalidate="true">
E-mail: <input type="email" name="user_email" />
<input type="submit" />
</form>
```

pattern 属性

pattern 属性规定用于验证 input 域的模式（pattern）。

模式（pattern） 是正则表达式。您可以在我们的 JavaScript 教程中学习到有关正则表达式的内容。

注释：pattern 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

下面的例子显示了一个只能包含三个字母的文本域（不含数字及特殊字符）：

实例

```
Country code: <input type="text" name="country_code"
pattern="[A-z]{3}" title="Three letter country code" />
```

placeholder 属性

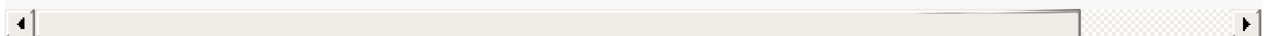
placeholder 属性提供一种提示（hint），描述输入域所期待的值。

注释：placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。

提示（hint）会在输入域为空时显示出现，会在输入域获得焦点时消失：

实例

```
<input type="search" name="user_search"  placeholder="Search W3Sch
```



required 属性

required 属性规定必须在提交之前填写输入域（不能为空）。

注释：required 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, number, checkbox, radio 以及 file。

实例

```
Name: <input type="text" name="usr_name" required="required" />
```

CSS 样式

CSS 背景

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。

CSS 在这方面的能力远远在 **HTML** 之上。

背景色

可以使用 background-color 属性为元素设置背景色。这个属性接受任何合法的颜色值。

这条规则把元素的背景设置为灰色：

```
p {background-color: gray;}
```

如果您希望背景色从元素中的文本向外少有延伸，只需增加一些内边距：

```
p {background-color: gray; padding: 20px;}
```

如需查看本例的效果，可以！

可以为所有元素设置背景色，这包括 body 一直到 em 和 a 等行内元素。

background-color 不能继承，其默认值是 transparent。transparent 有“透明”之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其祖先元素的背景才能可见。

背景图像

要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 URL 值：

```
body {background-image: url(img/eg_bg_04.gif);}
```

大多数背景都应用到 body 元素，不过并不仅限于此。

下面例子为一个段落应用了一个背景，而不会对文档的其他部分应用背景：

```
p.flower {background-image: url(img/eg_bg_03.gif);}
```

您甚至可以为行内元素设置背景图像，下面的例子为一个链接设置了背景图像：

```
a.radio {background-image: url(img/eg_bg_07.gif);}
```

如需查看上述例子的效果，可以！

理论上讲，甚至可以向 textareas 和 select 等替换元素的背景应用图像，不过并不是所有用户代理都能很好地处理这种情况。

另外还要补充一点，background-image 也不能继承。事实上，所有背景属性都不能继承。

背景重复

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性。

属性值 repeat 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺。

默认地，背景图像将从一个元素的左上角开始。请看下面的例子：

```
body
  {
  background-image: url(img/eg_bg_03.gif);
  background-repeat: repeat-y;
  }
```

如需查看上例的效果，可以。

背景定位

可以利用 background-position 属性改变图像在背景中的位置。

下面的例子在 body 元素中将一个背景图像居中放置：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:center;
  }
```

为 background-position 属性提供值有很多方法。首先，可以使用一些关键字：top、bottom、left、right 和 center。通常，这些关键字会成对出现，不过也不总是这样。还可以使用长度值，如 100px 或 5cm，最后也可以使用百分数值。不同类型的值对于背景图像的放置稍有差异。

关键字

图像放置关键字最容易理解，其作用如其名称所表明的。例如，top right 使图像放置在元素内边距区的右上角。

根据规范，位置关键字可以按任何顺序出现，只要保证不超过两个关键字 - 一个对应水平方向，另一个对应垂直方向。

如果只出现一个关键字，则认为另一个关键字是 center。

所以，如果希望每个段落的中部上方出现一个图像，只需声明如下：

```
p
  {
    background-image:url('bgimg.gif');
    background-repeat:no-repeat;
    background-position:top;
  }
```

下面是等价的位置关键字：

单一关键字	等价的关键字
center	center center
top	top center 或 center top
bottom	bottom center 或 center bottom
right	right center 或 center right
left	left center 或 center left

百分数值

百分数值的表现方式更为复杂。假设你希望用百分数值将图像在其元素中居中，这很容易：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50% 50%;
  }
```

这会导致图像适当放置，其中心与其元素的中心对齐。换句话说，百分数值同时应用于元素和图像。也就是说，图像中描述为 50% 50% 的点（中心点）与元素中描述为 50% 50% 的点（中心点）对齐。

如果图像位于 0% 0%，其左上角将放在元素内边距区的左上角。如果图像位置是 100% 100%，会使图像的右下角放在右边距的右下角。

因此，如果你想把一个图像放在水平方向 2/3、垂直方向 1/3 处，可以这样声明：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:66% 33%;
  }
```

如果只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 50%。这一点与关键字类似。

background-position 的默认值是 0% 0%，在功能上相当于 top left。这就解释了背景图像为什么总是从元素内边距区的左上角开始平铺，除非您设置了不同的位置值。

长度值

长度值解释的是元素内边距区左上角的偏移。偏移点是图像的左上角。

比如，如果设置值为 50px 100px，图像的左上角将在元素内边距区左上角向右 50 像素、向下 100 像素的位置上：

```
body
  {
    background-image:url('/i/eg_bg_03.gif');
    background-repeat:no-repeat;
    background-position:50px 100px;
  }
```

注意，这一点与百分数值不同，因为偏移只是从一个左上角到另一个左上角。也就是说，图像的左上角与 background-position 声明中的指定的点对齐。

背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。

您可以通过 background-attachment 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的（fixed），因此不会受到滚动的影响：

```
body
  {
  background-image:url(img/eg_bg_02.gif);
  background-repeat:no-repeat;
  background-attachment:fixed
  }
```

如需查看上例的效果，可以。

background-attachment 属性的默认值是 scroll，也就是说，在默认的情况下，背景会随文档滚动。

CSS 背景实例

设置背景颜色

设置文本的背景颜色

将图像设置为背景

将图像设置为背景 2

如何重复背景图像

如何在垂直方向重复背景图像

如何在水平方向重复背景图像

如何仅显示一次背景图像

如何放置背景图像

如何使用%来定位背景图像

如何使用像素来定位背景图像

如何设置固定的背景图像

所有背景属性在一个声明之中

CSS 背景属性

属性	描述
background	简写属性，作用是将背景属性设置在一个声明中。
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。
background-color	设置元素的背景颜色。
background-image	把图像设置为背景。
background-position	设置背景图像的起始位置。
background-repeat	设置背景图像是否及如何重复。

CSS 文本

CSS 文本属性可定义文本的外观。

通过文本属性，您可以改变文本的颜色、字符间距，对齐文本，装饰文本，对文本进行缩进，等等。

缩进文本

把 Web 页面上的段落的第一行缩进，这是一种最常用的文本格式化效果。

CSS 提供了 text-indent 属性，该属性可以方便地实现文本缩进。

通过使用 text-indent 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。

这个属性最常见的用途是将段落的首行缩进，下面的规则会使所有段落的首行缩进 5 em：

```
p {text-indent: 5em;}
```

注意：一般来说，可以为所有块级元素应用 text-indent，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 text-indent 属性。不过，如果一个块级元素（比如段落）的首行中有一个图像，它会随该行的其余文本移动。

提示：如果想把一个行内元素的第一行“缩进”，可以用左内边距或外边距创造这种效果。

使用负值

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的效果，比如“悬挂缩进”，即第一行悬挂在元素中余下部分的左边：

```
p {text-indent: -5em;}
```

不过在为 text-indent 设置负值时要当心，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距：

```
p {text-indent: -5em; padding-left: 5em;}
```

使用百分比值

text-indent 可以使用所有长度单位，包括百分比值。

百分数要相对于缩进元素父元素的宽度。换句话说，如果将缩进值设置为 20%，所影响元素的第一行会缩进其父元素宽度的 20%。

在下例中，缩进值是父元素的 20%，即 100 个像素：

```
div {width: 500px;}
p {text-indent: 20%;}

<div>
<p>this is a paragragh</p>
</div>
```

继承

text-indent 属性可以继承，请考虑如下标记：

```
div#outer {width: 500px;}
div#inner {text-indent: 10%;}
p {width: 200px;}

<div id="outer">
<div id="inner">some text. some text. some text.
<p>this is a paragragh.</p>
</div>
</div>
```

以上标记中的段落也会缩进 50 像素，这是因为这个段落继承了 id 为 inner 的 div 元素的缩进值。

水平对齐

text-align 是一个基本的属性，它会影响一个元素中的文本行互相之间的对齐方式。它的前 3 个值相当直接，不过第 4 个和第 5 个则略有些复杂。

值 left、right 和 center 会导致元素中的文本分别左对齐、右对齐和居中。

西方语言都是从左向右读，所有 text-align 的默认值是 left。文本在左边界对齐，右边界呈锯齿状（称为“从左到右”文本）。对于希伯来语和阿拉伯语之类的的语言，text-align 则默认为 right，因为这些语言从右向左读。不出所料，center 会使每个文本行在元素中居中。

提示：将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

text-align:center 与 <CENTER>

您可能会认为 text-align:center 与 <CENTER> 元素的作用一样，但实际上二者大不相同。

<CENTER> 不仅影响文本，还会把整个元素居中。text-align 不会控制元素的对齐，而只影响内部内容。元素本身不会从一段移到另一端，只是其中的文本受影响。

justify

最后一个水平对齐属性是 justify。

在两端对齐文本中，文本行的左右两端都放在父元素的内边界上。然后，调整单词和字母间的间隔，使各行的长度恰好相等。您也许已经注意到了，两端对齐文本在打印领域很常见。

需要注意的是，要由用户代理（而不是 CSS）来确定两端对齐文本如何拉伸，以填满父元素左右边界之间的空间。如需了解详情，请参阅 CSS text-align 属性参考页。

字间隔

word-spacing 属性可以改变字（单词）之间的标准间隔。其默认值 normal 与设置值为 0 是一样的。

word-spacing 属性接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把它拉近：

```
p.spread {word-spacing: 30px;}
p.tight {word-spacing: -0.5em;}

<p class="spread">
This is a paragraph. The spaces between words will be increased.
</p>

<p class="tight">
This is a paragraph. The spaces between words will be decreased.
</p>
```

实例 TIY ：增加或减少单词间距（字间隔）

注释：如需深入理解 CSS 对“字”（word）的定义，请访问 CSS word-spacing 属性参考页。

字母间隔

letter-spacing 属性与 word-spacing 的区别在于，字母间隔修改的是字符或字母之间的间隔。

与 word-spacing 属性一样，letter-spacing 属性的可取值包括所有长度。默认关键字是 normal（这与 letter-spacing:0 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量：

```
h1 {letter-spacing: -0.5em}
h4 {letter-spacing: 20px}

<h1>This is header 1</h1>
<h4>This is header 4</h4>
```

实例 TIY ：规定字符间距（字母间隔）

字符转换

text-transform 属性处理文本的大小写。这个属性有 4 个值：

- none
- uppercase
- lowercase
- capitalize

默认值 none 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。最后，capitalize 只对每个单词的首字母大写。

作为一个属性，text-transform 可能无关紧要，不过如果您突然决定把所有 h1 元素变为大写，这个属性就很有用。不必单独地修改所有 h1 元素的内容，只需使用 text-transform 为你完成这个修改：

```
h1 {text-transform: uppercase}
```

使用 text-transform 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无需修改 h1 元素本身。其次，如果您以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

实例 TIY ：控制文本中字母的大小写

文本装饰

接下来，我们讨论 text-decoration 属性，这是一个很有意思的属性，它提供了很多非常有趣的行为。

text-decoration 有 5 个值：

- none
- underline
- overline
- line-through
- blink

不出所料，underline 会对元素加下划线，就像 HTML 中的 U 元素一样。overline 的作用恰好相反，会在文本的顶端画一个上划线。值 line-through 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 元素。blink 会让文本闪烁，类似于 Netscape 支持的颇招非议的 blink 标记。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。例如，链接默认地会有下划线。如果您希望去掉超链接的下划线，可以使用以下 CSS 来做到这一点：

```
a {text-decoration: none;}
```

注意：如果显式地用这样一个规则去掉链接的下划线，那么锚与正常文本之间在视觉上的唯一差别就是颜色（至少默认是这样的，不过也不能完全保证其颜色肯定有区别）。

还可以在一个规则中结合多种装饰。如果希望所有超链接既有下划线，又有上划线，则规则如下：

```
a:link a:visited {text-decoration: underline overline;}
```

不过要注意的是，如果两个不同的装饰都与同一元素匹配，胜出规则的值会完全取代另一个值。请考虑以下的规则：

```
h2.stricken {text-decoration: line-through;}
h2 {text-decoration: underline overline;}
```

对于给定的规则，所有 class 为 stricken 的 h2 元素都只有一个贯穿线装饰，而没有下划线和上划线，因为 *text-decoration* 值会替换而不是累积起来。

处理空白符

white-space 属性会影响到用户代理对源文档中的空格、换行和 tab 字符的处理。

通过使用该属性，可以影响浏览器处理字之间和文本行之间的空白符的方式。从某种程度上讲，默认的 XHTML 处理已经完成了空白符处理：它会把所有空白符合并为一个空格。所以给定以下标记，它在 Web 浏览器中显示时，各个字之间只会显示一个空格，同时忽略元素中的换行：

```
<p>This     paragraph has    many
    spaces           in it.</p>
```

可以用以下声明显式地设置这种默认行为：

```
p {white-space: normal;}
```

上面的规则告诉浏览器按照平常的做法去处理：丢掉多余的空白符。如果给定这个值，换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。

实例 TIY ：white-space: normal

值 pre

不过，如果将 white-space 设置为 pre，受这个属性影响的元素中，空白符的处理就有所不同，其行为就像 XHTML 的 pre 元素一样；空白符不会被忽略。

如果 white-space 属性的值为 pre，浏览器将会注意额外的空格，甚至回车。在这个方面，而且仅在这个方面，任何元素都可以相当于一个 pre 元素。

实例 TIY ：white-space: pre

注意：经测试，IE 7 以及更早版本的浏览器不支持该值，因此请使用非 IE 的浏览器来查看上面的实例。

值 nowrap

与之相对的值是 nowrap，它会防止元素中的文本换行，除非使用了一个 br 元素。在 CSS 中使用 nowrap 非常类似于 HTML 4 中用 <td nowrap> 将一个表单元格设置为不能换行，不过 white-space 值可以应用到任何元素。

实例 TIY ：white-space: nowrap

值 pre-wrap 和 pre-line

CSS2.1 引入了值 pre-wrap 和 pre-line，这在以前版本的 CSS 中是没有的。这些值的作用是允许创作人员更好地控制空白符处理。

如果元素的 white-space 设置为 pre-wrap，那么该元素中的文本会保留空白符序列，但是文本行会正常地换行。如果设置为这个值，源文本中的行分隔符以及生成的行分隔符也会保留。pre-line 与 pre-wrap 相反，会像正常文本中一样合并空白符序列，但保留换行符。

实例 TIY ：white-space: pre-wrap

实例 TIY ：white-space: pre-line

注意：我们在 IE7 和 FireFox2.0 浏览器中测试了上面的两个实例，但是结果是，值 pre-wrap 和 pre-line 都没有得到很好的支持。

总结

下面的表格总结了 white-space 属性的行为：

值	空白符	换行符	自动换行
pre-line	合并	保留	允许
normal	合并	忽略	允许
nowrap	合并	忽略	不允许
pre	保留	保留	不允许
pre-wrap	保留	保留	允许

文本方向

如果您阅读的是英文书籍，就会从左到右、从上到下地阅读，这就是英文的流方向。不过，并不是所有语言都如此。我们知道古汉语就是从右到左来阅读的，当然还包括希伯来语和阿拉伯语等等。CSS2 引入了一个属性来描述其方向性。

direction 属性影响块级元素中文本的书写方向、表中列布局的方向、内容水平填充其元素框的方向、以及两端对齐元素中最后一行的位置。

注释：对于行内元素，只有当 unicode-bidi 属性设置为 embed 或 bidi-override 时才会应用 direction 属性。

direction 属性有两个值：ltr 和 rtl。大多数情况下，默认值是 ltr，显示从左到右的文本。如果显示从右到左的文本，应使用值 rtl。

CSS 文本实例：

设置文本颜色

设置文本的背景颜色

规定字符间距

使用百分比设置行间距

使用像素值设置行间距

使用数值来设置行间距

对齐文本

修饰文本

缩进文本

控制文本中的字母

在元素中禁止文本折行

增加单词间距

CSS 文本属性

属性	描述
color	设置文本颜色
direction	设置文本方向。
line-height	设置行高。
letter-spacing	设置字符间距。
text-align	对齐元素中的文本。
text-decoration	向文本添加修饰。
text-indent	缩进元素中文本的首行。
text-shadow	设置文本阴影。CSS2 包含该属性，但是 CSS2.1 没有保留该属性。
text-transform	控制元素中的字母。
unicode-bidi	设置文本方向。
white-space	设置元素中空白的处理方式。
word-spacing	设置字间距。

CSS 字体

CSS 字体属性定义文本的字体系列、大小、加粗、风格（如斜体）和变形（如小型大写字母）。

CSS 字体系列

在 CSS 中，有两种不同类型的字体系列名称：

- 通用字体系列 - 拥有相似外观的字体系统组合（比如 "Serif" 或 "Monospace"）
- 特定字体系列 - 具体的字体系列（比如 "Times" 或 "Courier"）

除了各种特定的字体系列外，CSS 定义了 5 种通用字体系列：

- Serif 字体
- Sans-serif 字体
- Monospace 字体
- Cursive 字体
- Fantasy 字体

如果需要了解更多有关字体系列的知识，请阅读 CSS 字体系列。

指定字体系列

使用 font-family 属性 定义文本的字体系列。

使用通用字体系列

如果你希望文档使用一种 sans-serif 字体，但是你并不关心是哪一种字体，以下就是一个合适的声明：

```
body {font-family: sans-serif;}
```

这样用户代理就会从 sans-serif 字体系列中选择一个字体（如 Helvetica），并将其应用到 body 元素。因为有继承，这种字体选择还将应用到 body 元素中包含的所有元素，除非有一种更特定的选择器将其覆盖。

指定字体系列

除了使用通用的字体系列，您还可以通过 font-family 属性设置更具体的字体。

下面的例子为所有 h1 元素设置了 Georgia 字体：

```
h1 {font-family: Georgia;}
```

这样的规则同时会产生另外一个问题，如果用户代理上没有安装 Georgia 字体，就只能使用用户代理的默认字体来显示 h1 元素。

我们可以通过结合特定字体名和通用字体系列来解决这个问题：

```
h1 {font-family: Georgia, serif;}
```

如果读者没有安装 Georgia，但安装了 Times 字体（serif 字体系列中的一种字体），用户代理就可能对 h1 元素使用 Times。尽管 Times 与 Georgia 并不完全匹配，但至少足够接近。

因此，我们建议在所有 font-family 规则中都提供一个通用字体系列。这样就提供了一条后路，在用户代理无法提供与规则匹配的特定字体时，就可以选择一个候选字体。

如果您对字体非常熟悉，也可以为给定的元素指定一系列类似的字体。要做到这一点，需要把这些字体按照优先顺序排列，然后用逗号进行连接：

```
p {font-family: Times, TimesNR, 'New Century Schoolbook',
     Georgia, 'New York', serif;}
```

根据这个列表，用户代理会按所列的顺序查找这些字体。如果列出的所有字体都不可用，就会简单地选择一种可用的 serif 字体。

使用引号

您也许已经注意到了，上面的例子中使用了单引号。只有当字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号。

单引号或双引号都可以接受。但是，如果把一个 font-family 属性放在 HTML 的 style 属性中，则需要使用该属性本身未使用的那种引号：

```
<p style="font-family: Times, TimesNR, 'New Century Schoolbook', Ge
 'New York', serif;">...</p>
```

字体风格

font-style 属性最常用于规定斜体文本。

该属性有三个值：

- normal - 文本正常显示
- italic - 文本斜体显示
- oblique - 文本倾斜显示

实例

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

italic 和 oblique 的区别

font-style 非常简单：用于在 normal 文本、italic 文本和 oblique 文本之间选择。唯一有点复杂的是明确 italic 文本和 oblique 文本之间的差别。

斜体（italic）是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观。与此不同，倾斜（oblique）文本则是正常竖直文本的一个倾斜版本。

通常情况下，italic 和 oblique 文本在 web 浏览器中看上去完全一样。

字体变形

font-variant 属性可以设定小型大写字母。

小型大写字母不是一般的大写字母，也不是小写字母，这种字母采用不同大小的大写字母。

实例

```
p {font-variant:small-caps;}
```

字体加粗

font-weight 属性设置文本的粗细。

使用 bold 关键字可以将文本设置为粗体。

关键字 100 ~ 900 为字体指定了 9 级加粗度。如果一个字体内置了这些加粗级别，那么这些数字就直接映射到预定义的级别，100 对应最细的字体变形，900 对应最粗的字体变形。数字 400 等价于 normal，而 700 等价于 bold。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度下移而不是上移。

实例

```
p.normal {font-weight:normal;}
p.thick {font-weight:bold;}
p.thicker {font-weight:900;}
```

字体大小

font-size 属性设置文本的大小。

有能力管理文本的大小在 web 设计领域很重要。但是，您不应当通过调整文本大小使段落看上去像标题，或者使标题看上去像段落。

请始终使用正确的 HTML 标题，比如使用 <h1> - <h6> 来标记标题，使用 <p> 来标记段落。

font-size 值可以是绝对或相对值。

绝对值：

- 将文本设置为指定的大小
- 不允许用户在所有浏览器中改变文本大小（不利于可用性）
- 绝对大小在确定了输出的物理尺寸时很有用

相对大小：

- 相对于周围的元素来设置大小
- 允许用户在浏览器改变文本大小

注意：如果您没有规定字体大小，普通文本（比如段落）的默认大小是 16 像素 (16px=1em)。

使用像素来设置字体大小

通过像素设置文本大小，可以对文本大小进行完全控制：

实例

```
h1 {font-size:60px;}
h2 {font-size:40px;}
p {font-size:14px;}
```

在 Firefox, Chrome, and Safari 中，可以重新调整以上例子的文本大小，但是在 Internet Explorer 中不行。

虽然可以通过浏览器的缩放工具调整文本大小，但是这实际上是对整个页面的调整，而不仅限于文本。

使用 em 来设置字体大小

如果要避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels。

W3C 推荐使用 em 尺寸单位。

1em 等于当前的字体尺寸。如果一个元素的 font-size 为 16 像素，那么对于该元素，1em 就等于 16 像素。在设置字体大小时，em 的值会相对于父元素的字体大小改变。

浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。

可以使用下面这个公式将像素转换为 em：*pixels*/16=*em*

（注：16 等于父元素的默认字体大小，假设父元素的 font-size 为 20px，那么公式需改为：*pixels*/20=*em*）

实例

```
h1 {font-size:3.75em;} /* 60px/16=3.75em */
h2 {font-size:2.5em;}  /* 40px/16=2.5em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

在上面的例子中，以 em 为单位的文本大小与前一个例子中以像素计的文本是相同的。不过，如果使用 em 单位，则可以在所有浏览器中调整文本大小。

不幸的是，在 IE 中仍存在问题。在重设文本大小时，会比正常的尺寸更大或更小。

结合使用百分比和 EM

在所有浏览器中均有效的方案是为 body 元素（父元素）以百分比设置默认的 font-size 值：

实例

```
body {font-size:100%;}
h1 {font-size:3.75em;}
h2 {font-size:2.5em;}
p {font-size:0.875em;}
```

我们的代码非常有效。在所有浏览器中，可以显示相同的文本大小，并允许所有浏览器缩放文本的大小。

CSS 字体实例：

设置文本的字体

设置字体尺寸

设置字体风格

设置字体的异体

设置字体的粗细

所有字体属性在一个声明之内

CSS 字体属性

属性	描述
font	简写属性。作用是把所有针对字体的属性设置在一个声明中。
font-family	设置字体系列。
font-size	设置字体的尺寸。
font-size-adjust	当首选字体不可用时，对替换字体进行智能缩放。（CSS2.1 已删除该属性。）
font-stretch	对字体进行水平拉伸。（CSS2.1 已删除该属性。）
font-style	设置字体风格。
font-variant	以小型大写字体或者正常字体显示文本。
font-weight	设置字体的粗细。

CSS 链接

我们能够以不同的方法为链接设置样式。

设置链接的样式

能够设置链接样式的 CSS 属性有很多种（例如 color, font-family, background 等等）。

链接的特殊性在于能够根据它们所处的状态来设置它们的样式。

链接的四种状态：

- a:link - 普通的、未被访问的链接
- a:visited - 用户已访问的链接
- a:hover - 鼠标指针位于链接的上方
- a:active - 链接被点击的时刻

实例

```
a:link {color:#FF0000;}        /* 未被访问的链接 */
a:visited {color:#00FF00;}    /* 已被访问的链接 */
a:hover {color:#FF00FF;}    /* 鼠标指针移动到链接上 */
a:active {color:#0000FF;}    /* 正在被点击的链接 */
```

当为链接的不同状态设置样式时，请按照以下次序规则：

- a:hover 必须位于 a:link 和 a:visited 之后
- a:active 必须位于 a:hover 之后

常见的链接样式

在上面的例子中，链接根据其状态改变颜色。

让我们看看其他几种常见的设置链接样式的方法：

文本修饰

text-decoration 属性大多用于去掉链接中的下划线：

实例

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

背景色

background-color 属性规定链接的背景色：

实例

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

更多实例

向链接添加不同的样式

高级 - 创建链接框

CSS 列表

CSS 列表属性允许你放置、改变列表项标志，或者将图像作为列表项标志。

CSS 列表

从某种意义上讲，不是描述性的文本的任何内容都可以认为是列表。人口普查、太阳系、家谱、参观菜单，甚至你的所有朋友都可以表示为一个列表或者是列表的列表。

由于列表如此多样，这使得列表相当重要，所以说，CSS 中列表样式不太丰富确实是一大憾事。

列表类型

要影响列表的样式，最简单（同时支持最充分）的办法就是改变其标志类型。

例如，在一个无序列表中，列表项的标志 (marker) 是出现在各列表项旁边的圆点。在有序列表中，标志可能是字母、数字或另外某种计数体系中的一个符号。

要修改用于列表项的标志类型，可以使用属性 list-style-type：

```
ul {list-style-type : square}
```

上面的声明把无序列表中的列表项标志设置为方块。

列表项图像

有时，常规的标志是不够的。你可能想对各标志使用一个图像，这可以利用 list-style-image 属性做到：

```
ul li {list-style-image : url(xxx.gif)}
```

只需要简单地使用一个 url() 值，就可以使用图像作为标志。

列表标志位置

CSS2.1 可以确定标志出现在列表项内容之外还是内容内部。这是利用 list-style-position 完成的。

简写列表样式

为简单起见，可以将以上 3 个列表样式属性合并为一个方便的属性：list-style，就像这样：

```
li {list-style : url(example.gif) square inside}
```

list-style 的值可以按任何顺序列出，而且这些值都可以忽略。只要提供了一个值，其它的就会填入其默认值。

CSS 列表实例：

在无序列表中的不同类型的列表标记

在有序列表中不同类型的列表项标记

所有的列表样式类型

将图像作为列表项标记

放置列表标记

在一个声明中定义所有的列表属性

CSS 列表属性(list)

属性	描述
list-style	简写属性。用于把所有用于列表的属性设置于一个声明中。
list-style-image	将图象设置为列表项标志。
list-style-position	设置列表中列表项标志的位置。
list-style-type	设置列表项标志的类型。
marker-offset	

CSS 表格

CSS 表格属性可以帮助您极大地改善表格的外观。

表格边框

如需在 CSS 中设置表格边框，请使用 border 属性。

下面的例子为 table、th 以及 td 设置了蓝色边框：

```
table, th, td
  {
  border: 1px solid blue;
  }
```

请注意，上例中的表格具有双线条边框。这是由于 table、th 以及 td 元素都有独立的边框。

如果需要把表格显示为单线条边框，请使用 border-collapse 属性。

折叠边框

border-collapse 属性设置是否将表格边框折叠为单一边框：

```
table
  {
  border-collapse:collapse;
  }

table,th, td
  {
  border: 1px solid black;
  }
```

表格宽度和高度

通过 width 和 height 属性定义表格的宽度和高度。

下面的例子将表格宽度设置为 100%，同时将 th 元素的高度设置为 50px：

```
table
  {
  width:100%;
  }

th
  {
  height:50px;
  }
```

表格文本对齐

text-align 和 vertical-align 属性设置表格中文本的对齐方式。

text-align 属性设置水平对齐方式，比如左对齐、右对齐或者居中：

```
td
  {
  text-align:right;
  }
```

vertical-align 属性设置垂直对齐方式，比如顶部对齐、底部对齐或居中对齐：

```
td
  {
  height:50px;
  vertical-align:bottom;
  }
```

表格内边距

如需控制表格中内容与边框的距离，请为 td 和 th 元素设置 padding 属性：

```
td
  {
  padding:15px;
  }
```

表格颜色

下面的例子设置边框的颜色，以及 th 元素的文本和背景颜色：

```
table, td, th
  {
  border:1px solid green;
  }

th
  {
  background-color:green;
  color:white;
  }
```

CSS Table 属性

属性	描述
border-collapse	设置是否把表格边框合并为单一的边框。
border-spacing	设置分隔单元格边框的距离。
caption-side	设置表格标题的位置。
empty-cells	设置是否显示表格中的空单元格。
table-layout	设置显示单元、行和列的算法。

亲自试一试 - 更多实例

制作一个漂亮的表格

显示表格中的空单元

设置表格边框之间的空白

设置表格标题的位置

CSS 轮廓

轮廓（outline）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

CSS outline 属性规定元素轮廓的样式、颜色和宽度。

轮廓（Outline） 实例：

在元素周围画线

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border:red solid thin;
outline:#00ff00 dotted thick;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时，Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的颜色

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border:red solid thin;
outline-style:dotted;
outline-color:#00ff00;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时，Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的样式

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
border: red solid thin;
}
p.dotted {outline-style: dotted}
p.dashed {outline-style: dashed}
p.solid {outline-style: solid}
p.double {outline-style: double}
p.groove {outline-style: groove}
p.ridge {outline-style: ridge}
p.inset {outline-style: inset}
p.outset {outline-style: outset}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="dotted"&gt;A dotted outline&lt;/p&gt;
&lt;p class="dashed"&gt;A dashed outline&lt;/p&gt;
&lt;p class="solid"&gt;A solid outline&lt;/p&gt;
&lt;p class="double"&gt;A double outline&lt;/p&gt;
&lt;p class="groove"&gt;A groove outline&lt;/p&gt;
&lt;p class="ridge"&gt;A ridge outline&lt;/p&gt;
&lt;p class="inset"&gt;An inset outline&lt;/p&gt;
&lt;p class="outset"&gt;An outset outline&lt;/p&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时, Internet E
&lt;/body&gt;
&lt;/html&gt;
```

设置轮廓的宽度

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p.one
{
border:red solid thin;
outline-style:solid;
outline-width:thin;
}
p.two
{
border:red solid thin;
outline-style:dotted;
outline-width:3px;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="one"&gt;This is some text in a paragraph.&lt;/p&gt;
&lt;p class="two"&gt;This is some text in a paragraph.&lt;/p&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;只有在规定了 !DOCTYPE 时，Internet E
&lt;/body&gt;
&lt;/html&gt;
```

CSS 边框属性

"CSS" 列中的数字指示哪个 CSS 版本定义了该属性。

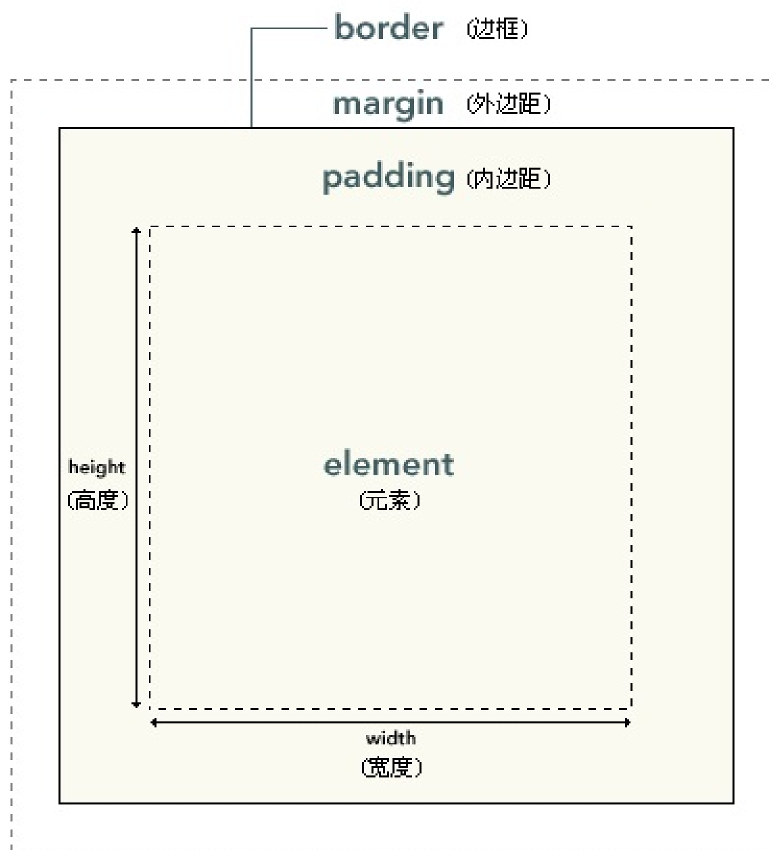
属性	描述	CSS
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	2

CSS 框模型

CSS 框模型概述

CSS 框模型 **(Box Model)** 规定了元素框处理元素内容、内边距、边框 和 外边距 的方式。

CSS 框模型概述



元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

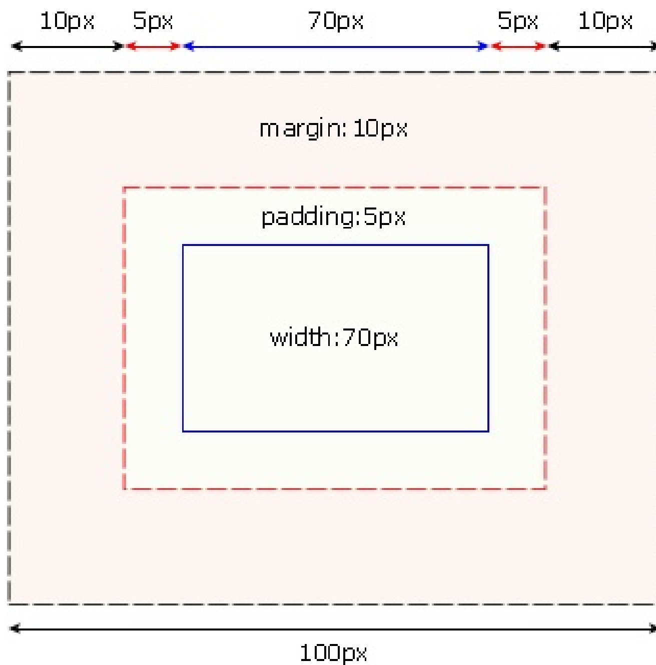
提示：背景应用于由内容和内边距、边框组成的区域。

内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

```
* {
  margin: 0;
  padding: 0;
}
```

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，请看下图：




```
#box {
  width: 70px;
  margin: 10px;
  padding: 5px;
}
```

提示：内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。

提示：外边距可以是负值，而且在很多情况下都要使用负值的外边距。

浏览器兼容性

一旦为页面设置了恰当的 DTD，大多数浏览器都会按照上面的图示来呈现内容。然而 IE 5 和 6 的呈现却是不正确的。根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的。不幸的是，IE5.X 和 6 在怪异模式中使用自己的非标准模型。这些浏览器的 width 属性不是内容的宽度，而是内容、内边距和边框的宽度的总和。

虽然有方法解决这个问题。但是目前最好的解决方案是回避这个问题。也就是，不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素和子元素。

术语翻译

- element : 元素。
- padding : 内边距，也有资料将其翻译为填充。
- border : 边框。
- margin : 外边距，也有资料将其翻译为空白或空白边。

在 w3school，我们把 padding 和 margin 统一地称为内边距和外边距。边框内的空白是内边距，边框外的空白是外边距，很容易记吧：）

CSS 内边距

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 **padding** 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性

CSS padding 属性定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。

例如，如果您希望所有 h1 元素的各边都有 10 像素的内边距，只需要这样：

```
h1 {padding: 10px;}
```

您还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
h1 {padding: 10px 0.25em 2ex 20%;}
```

单边内边距属性

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

- padding-top
- padding-right
- padding-bottom
- padding-left

您也许已经想到了，下面的规则实现的效果与上面的简写规则是完全相同的：

```
h1 {
  padding-top: 10px;
  padding-right: 0.25em;
  padding-bottom: 2ex;
  padding-left: 20%;
  }
```

内边距的百分比数值

前面提到过，可以为元素的内边距设置百分数值。百分数值是相对于其父元素的 width 计算的，这一点与外边距一样。所以，如果父元素的 width 改变，它们也会改变。

下面这条规则把段落的内边距设置为父元素 width 的 10%：

```
p {padding: 10%;}
```

例如：如果一个段落的父元素是 div 元素，那么它的内边距要根据 div 的 width 计算。

```
<div style="width: 200px;">
<p>This paragragh is contained within a DIV that has a width of 200
</div>
```

注意：上下内边距与左右内边距一致；即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

CSS 内边距实例：

所有内边距属性在一个声明中

设置下内边距 1

设置下内边距 2

设置左内边距 1

设置左内边距 2

设置右内边距 1

设置右内边距 2

设置上内边距 1

设置上内边距 2

CSS 内边距属性

属性	描述
padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
padding-bottom	设置元素的下内边距。
padding-left	设置元素的左内边距。
padding-right	设置元素的右内边距。
padding-top	设置元素的上内边距。

CSS 边框

元素的边框 (border) 是围绕元素内容和内边距的一条或多条线。

CSS border 属性允许你规定元素边框的样式、宽度和颜色。

CSS 边框

在 HTML 中，我们使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的的边框 (border)。元素的边框就是围绕元素内容和内边据的一条或多条线。

每个边框有 3 个方面：宽度、样式，以及颜色。在下面的篇幅，我们会为您详细讲解这三个方面。

边框与背景

CSS 规范指出，边框绘制在“元素的背景之上”。这很重要，因为有些边框是“间断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。

CSS2 指出背景只延伸到内边距，而不是边框。后来 CSS2.1 进行了更正：元素的背景是内容、内边距和边框区的背景。大多数浏览器都遵循 CSS2.1 定义，不过一些较老的浏览器可能会有不同的表现。

边框的样式

样式是边框最重要的一个方面，这不是因为样式控制着边框的显示（当然，样式确实控制着边框的显示），而是因为如果没有样式，将根本没有边框。

CSS 的 border-style 属性定义了 10 个不同的非 inherit 样式，包括 none。

例如，您可以为把一幅图片的边框定义为 outset，使之看上去像是“凸起按钮”：

```
a:link img {border-style: outset;}
```

定义多种样式

您可以为一个边框定义多个样式，例如：

```
p.aside {border-style: solid dotted dashed double;}
```

上面这条规则为类名为 aside 的段落定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。

我们又看到了这里的值采用了 top-right-bottom-left 的顺序，讨论用多个值设置不同内边距时也见过这个顺序。

定义单边样式

如果您希望为元素框的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用下面的单边边框样式属性：

- border-top-style
- border-right-style
- border-bottom-style
- border-left-style

因此这两种方法是等价的：

```
p {border-style: solid solid solid none;}
p {border-style: solid; border-left-style: none;}
```

注意：如果要使用第二种方法，必须把单边属性放在简写属性之后。因为如果把单边属性放在 border-style 之前，简写属性的值就会覆盖单边值 none。

边框的宽度

您可以通过 border-width 属性为边框指定宽度。

为边框指定宽度有两种方法：可以指定长度值，比如 2px 或 0.1em；或者使用 3 个关键字之一，它们分别是 thin 、medium（默认值） 和 thick。

注释：CSS 没有定义 3 个关键字的具体宽度，所以一个用户代理可能把 thin 、medium 和 thick 分别设置为等于 5px、3px 和 2px，而另一个用户代理则分别设置为 3px、2px 和 1px。

所以，我们可以这样设置边框的宽度：

```
p {border-style: solid; border-width: 5px;}
```

或者：

```
p {border-style: solid; border-width: thick;}
```

定义单边宽度

您可以按照 top-right-bottom-left 的顺序设置元素的各边边框：

```
p {border-style: solid; border-width: 15px 5px 15px 5px;}
```

上面的例子也可以简写为（这样写法称为值复制）：

```
p {border-style: solid; border-width: 15px 5px;}
```

您也可以通过下列属性分别设置边框各边的宽度：

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width

因此，下面的规则与上面的例子是等价的：

```
p {
  border-style: solid;
  border-top-width: 15px;
  border-right-width: 5px;
  border-bottom-width: 15px;
  border-left-width: 5px;
  }
```

没有边框

在前面的例子中，您已经看到，如果希望显示某种边框，就必须设置边框样式，比如 solid 或 outset。

那么如果把 border-style 设置为 none 会出现什么情况：

```
p {border-style: none; border-width: 50px;}
```

尽管边框的宽度是 50px，但是边框样式设置为 none。在这种情况下，不仅边框的样式没有了，其宽度也会变成 0。边框消失了，为什么呢？

这是因为如果边框样式为 none，即边框根本不存在，那么边框就不可能有宽度，因此边框宽度自动设置为 0，而不论您原先定义的是什么。

记住这一点非常重要。事实上，忘记声明边框样式是一个常犯的错误。根据以下规则，所有 h1 元素都不会有任何边框，更不用说 20 像素宽了：

```
h1 {border-width: 20px;}
```

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。因此，如果您希望边框出现，就必须声明一个边框样式。

边框的颜色

设置边框颜色非常简单。CSS 使用一个简单的 border-color 属性，它一次可以接受最多 4 个颜色值。

可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制和 RGB 值：

```
p {
  border-style: solid;
  border-color: blue rgb(25%,35%,45%) #909090 red;
  }
```

如果颜色值小于 4 个，值复制就会起作用。例如下面的规则声明了段落的上下边框是蓝色，左右边框是红色：

```
p {
  border-style: solid;
  border-color: blue red;
  }
```

注释：默认的边框颜色是元素本身的前景色。如果没有为边框声明颜色，它将与元素的文本颜色相同。另一方面，如果元素没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父元素的文本颜色（因为 color 可以继承）。这个父元素很可能是 body、div 或另一个 table。

定义单边颜色

还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

- border-top-color
- border-right-color
- border-bottom-color
- border-left-color

要为 h1 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

```
h1 {
  border-style: solid;
  border-color: black;
  border-right-color: red;
  }
```

透明边框

我们刚才讲过，如果边框没有样式，就没有宽度。不过有些情况下您可能希望创建一个不可见的边框。

CSS2 引入了边框颜色值 transparent。这个值用于创建有宽度的不可见边框。请看下面的例子：

```
<a href="#">AAA</a>
<a href="#">BBB</a>
<a href="#">CCC</a>
```

我们为上面的链接定义了如下样式：

```
a:link, a:visited {
  border-style: solid;
  border-width: 5px;
  border-color: transparent;
  }
a:hover {border-color: gray;}
```

如需查看以上样式的效果，请点击：。

从某种意义上说，利用 transparent，使用边框就像是额外的内边距一样；此外还有一个好处，就是能在你需要的时候使其可见。这种透明边框相当于内边距，因为元素的背景会延伸到边框区域（如果有可见背景的话）。

重要事项：在 IE7 之前，IE/WIN 没有提供对 transparent 的支持。在以前的版本，IE 会根据元素的 color 值来设置边框颜色。

CSS 边框实例：

所有边框属性在一个声明之中

设置四边框样式

设置每一边的不同边框

所有边框宽度属性在一个声明之中

设置四个边框的颜色

所有下边框属性在一个声明中

设置下边框的颜色

设置下边框的样式

设置下边框的宽度

所有左边框属性在一个声明之中

设置左边框的颜色

设置左边框的样式

设置左边框的宽度

所有右边框属性在一个声明之中

设置右边框的颜色

设置右边框的样式

设置右边框的宽度

所有上边框属性在一个声明之中

设置上边框的颜色

设置上边框的样式

设置上边框的宽度

CSS 边框属性

属性	描述
border	简写属性，用于把针对四个边的属性设置在一个声明。
border-style	用于设置元素所有边框的样式，或者单独地为各边设置边框样式。
border-width	简写属性，用于为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。
border-color	简写属性，设置元素的所有边框中可见部分的颜色，或为 4 个边分别设置颜色。
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中。
border-bottom-color	设置元素的下边框的颜色。
border-bottom-style	设置元素的下边框的样式。
border-bottom-width	设置元素的下边框的宽度。
border-left	简写属性，用于把左边框的所有属性设置到一个声明中。
border-left-color	设置元素的左边框的颜色。

border-left-style	设置元素的左边框的样式。
border-left-width	设置元素的左边框的宽度。
border-right	简写属性，用于把右边框的所有属性设置到一个声明中。
border-right-color	设置元素的右边框的颜色。
border-right-style	设置元素的右边框的样式。
border-right-width	设置元素的右边框的宽度。
border-top	简写属性，用于把上边框的所有属性设置到一个声明中。
border-top-color	设置元素的上边框的颜色。
border-top-style	设置元素的上边框的样式。
border-top-width	设置元素的上边框的宽度。

CSS 外边距

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距的最简单的方法就是使用 **margin** 属性，这个属性接受任何长度单位、百分数值甚至负值。

CSS margin 属性

设置外边距的最简单的方法就是使用 margin 属性。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 em。

margin 可以设置为 auto。更常见的做法是为外边距设置长度值。下面的声明在 h1 元素的各个边上设置了 1/4 英寸宽的空白：

```
h1 {margin : 0.25in;}
```

下面的例子为 h1 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)：

```
h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距 (top) 开始围着元素顺时针旋转的：

```
margin: top right bottom left
```

另外，还可以为 margin 设置一个百分比数值：

```
p {margin : 10%;}
```

百分数是相对于父元素的 width 计算的。上面这个例子为 p 元素设置的外边距是其父元素的 width 的 10%。

margin 的默认值是 0，所以如果没有为 margin 声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，外边距会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 p 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要你特别作了声明，就会覆盖默认样式。

值复制

还记得吗？我们曾经在前两节中提到过值复制。下面我们为您讲解如何使用值复制。

有时，我们会输入一些重复的值：

```
p {margin: 0.5em 1em 0.5em 1em;}
```

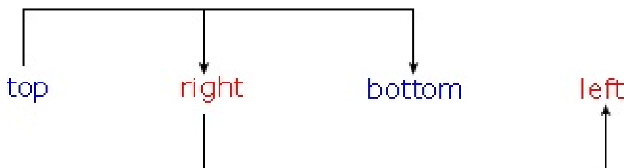
通过值复制，您可以不必重复地键入这对数字。上面的规则与下面的规则是等价的：

```
p {margin: 0.5em 1em;}
```

这两个值可以取代前面 4 个值。这是如何做到的呢？CSS 定义了一些规则，允许为外边距指定少于 4 个值。规则如下：

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

下图提供了更直观的方法来了解这一点：



换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到。

利用这个简单的机制，您只需指定必要的值，而不必全部都应用 4 个值，例如：

```
h1 {margin: 0.25em 1em 0.5em;}     /* 等价于 0.25em 1em 0.5em 1em */
h2 {margin: 0.5em 1em;}          /* 等价于 0.5em 1em 0.5em 1em */
p {margin: 1px;}               /* 等价于 1px 1px 1px 1px */
```

这种办法有一个小缺点，您最后肯定会遇到这个问题。假设希望把 p 元素的上外边距和左外边距设置为 20 像素，下外边距和右外边距设置为 30 像素。在这种情况下，必须写作：

```
p {margin: 20px 30px 30px 20px;}
```

这样才能得到您想要的结果。遗憾的是，在这种情况下，所需值的个数没有办法更少了。

再来看另外一个例子。如果希望除了左外边距以外所有其他外边距都是 auto（左外边距是 20px）：

```
p {margin: auto auto auto 20px;}
```

同样的，这样才能得到你想要的效果。问题在于，键入这些 auto 有些麻烦。如果您只是希望控制元素单边上的外边距，请使用单边外边距属性。

单边外边距属性

您可以使用单边外边距属性为元素单边上的外边距设置值。假设您希望把 p 元素的左外边距设置为 20px。不必使用 margin（需要键入很多 auto），而是可以采用以下方法：

```
p {margin-left: 20px;}
```

您可以使用下列任何一个属性来只设置相应上的外边距，而不会直接影响所有其他外边距：

- margin-top
- margin-right
- margin-bottom
- margin-left

一个规则中可以使用多个这种单边属性，例如：

```
h2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 30px;
  margin-left: 20px;
  }
```

当然，对于这种情况，使用 margin 可能更容易一些：

```
p {margin: 20px 30px 30px 20px;}
```

不论使用单边属性还是使用 margin，得到的结果都一样。一般来说，如果希望为多个边设置外边距，使用 margin 会更容易一些。不过，从文档显示的角度看，实际上使用哪种方法都不重要，所以应该选择对自己来说更容易的一种方法。

提示和注释

提示：Netscape 和 IE 对 body 标签定义的默认边距（margin）值是 8px。而 Opera 不是这样。相反地，Opera 将内部填充（padding）的默认值定义为 8px，因此如果希望对整个网站的边缘部分进行调整，并将之正确显示于 Opera 中，那么必须对 body 的 padding 进行自定义。

CSS 外边距实例：

设置文本的左外边距

设置文本的右外边距

设置文本的上外边距

设置文本的下外边距

所有的外边距属性在一个声明中。

CSS 外边距属性

属性	描述
margin	简写属性。在一个声明中设置所有外边距属性。
margin-bottom	设置元素的下外边距。
margin-left	设置元素的左外边距。
margin-right	设置元素的右外边距。
margin-top	设置元素的上外边距。

CSS 外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

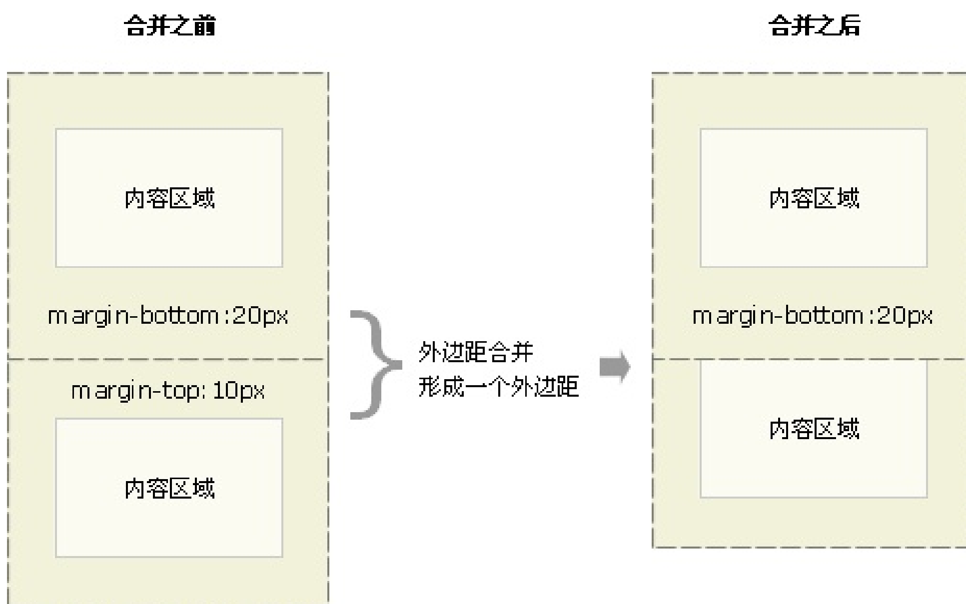
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

外边距合并

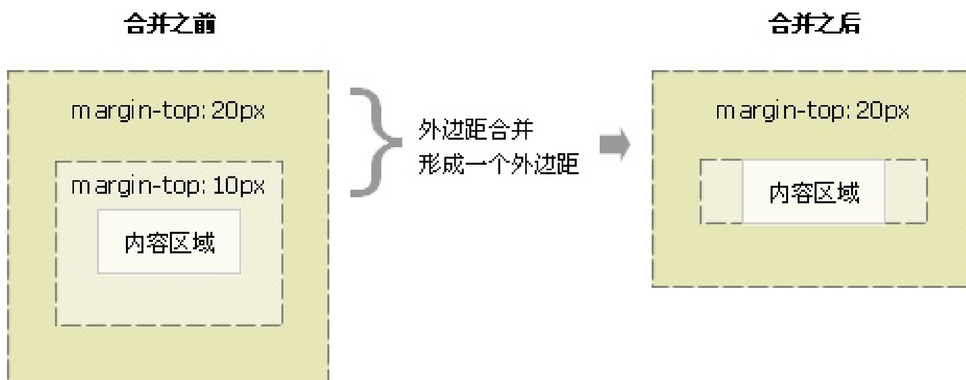
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

简单地说，外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。请看下图：

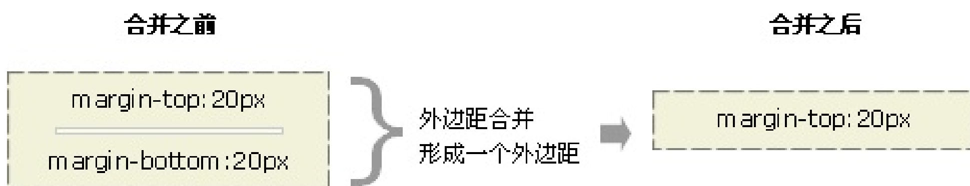


当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：

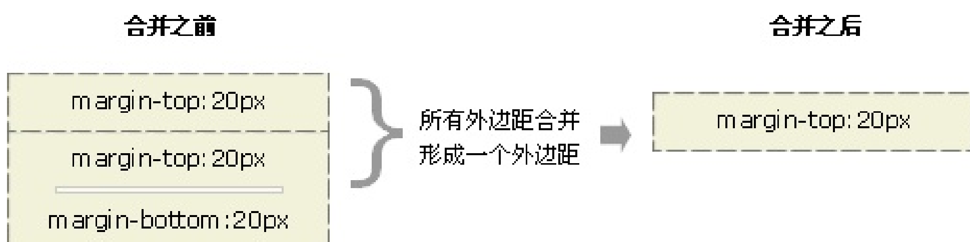


尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：

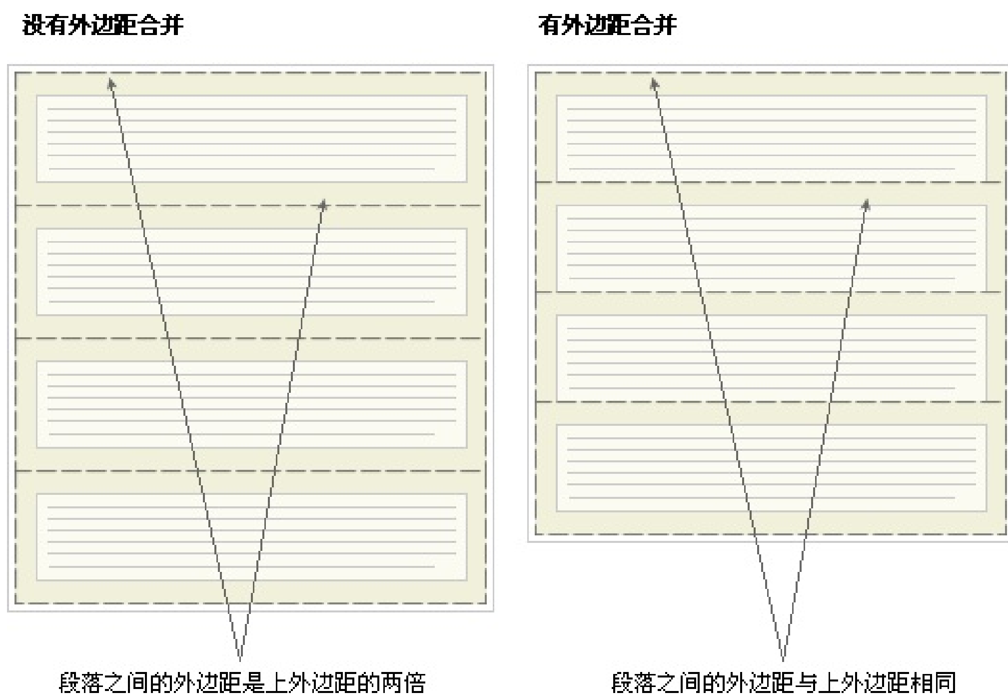


如果这个外边距遇到另一个元素的外边距，它还会发生合并：



这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

外边距合并初看上去可能有点奇怪，但是实际上，它是有意义的。以由几个段落组成的典型文本页面为例。第一个段落上面的空间等于段落的上外边距。如果没有外边距合并，后续所有段落之间的外边距都将是相邻上外边距和下外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距合并，段落之间的上外边距和下外边距就合并在一起，这样各处的距离就一致了。



注释：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

CSS 定位

CSS 定位 (Positioning)

CSS 定位 (Positioning) 属性允许你对元素进行定位。

CSS 定位和浮动

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。显然，这个功能非常强大，也很让人吃惊。要知道，用户代理对 CSS2 中定位的支持远胜于对其它方面的支持，对此不应感到奇怪。

另一方面，CSS1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

一切皆为框

div、h1 或 p 元素常常被称为块级元素。这意味着这些元素显示为一块内容，即“块框”。与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

您可以使用 display 属性改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如 <a> 元素）表现得像块级元素一样。还可以通过把 display 设置为 none，让生成的元素根本没有框。这样的话，该框及其所有内容就不再显示，不占用文档中的空间。

但是在一种情况下，即使没有进行显式定义，也会创建块级元素。这种情况发生在把一些文本添加到一个块级元素（比如 div）的开头。即使没有把这些文本定义为段落，它也会被当作段落对待：

```
<div>
some text
<p>Some more text.</p>
</div>
```

在这种情况下，这个框称为无名块框，因为它不与专门定义的元素相关联。

块级元素的文本行也会发生类似的情况。假设有一个包含三行文本的段落。每行文本形成一个无名框。无法直接对无名块或行框应用样式，因为没有可以应用样式的地方（注意，行框和行内框是两个概念）。但是，这有助于理解在屏幕上看到的所

有东西都形成某种框。

CSS 定位机制

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中的元素的位置由元素在 (X)HTML 中的位置决定。

块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来。

行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为行框（*Line Box*），行框的高度总是足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

在下面的章节，我们会为您详细讲解相对定位、绝对定位和浮动。

CSS position 属性

通过使用 position 属性，我们可以选择 4 种不同类型的定位，这会影响元素框生成的方式。

position 属性值的含义：

static

relative

absolute

fixed

提示：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

实例

定位：相对定位

定位：绝对定位

定位：固定定位

使用固定值设置图像的上边缘

使用百分比设置图像的上边缘

使用像素值设置图像的底部边缘

使用百分比设置图像的底部边缘

使用固定值设置图像的左边缘

使用百分比设置图像的左边缘

使用固定值设置图像的右边缘

使用百分比设置图像的右边缘

如何使用滚动条来显示元素内溢出的内容

如何隐藏溢出元素中溢出的内容

如何设置浏览器来自动地处理溢出

设置元素的形状

垂直排列图象

Z-index

Z-index

CSS 定位属性

CSS 定位属性允许你对元素进行定位。

属性	描述
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移。
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移。
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移。
left	定义了定位元素左外边距边界与其包含块左边界之间的偏移。
overflow	设置当元素的内容溢出其区域时发生的事情。
clip	设置元素的形状。元素被剪入这个形状之中，然后显示出来。
vertical-align	设置元素的垂直对齐方式。
z-index	设置元素的堆叠顺序。

CSS 相对定位

设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留。

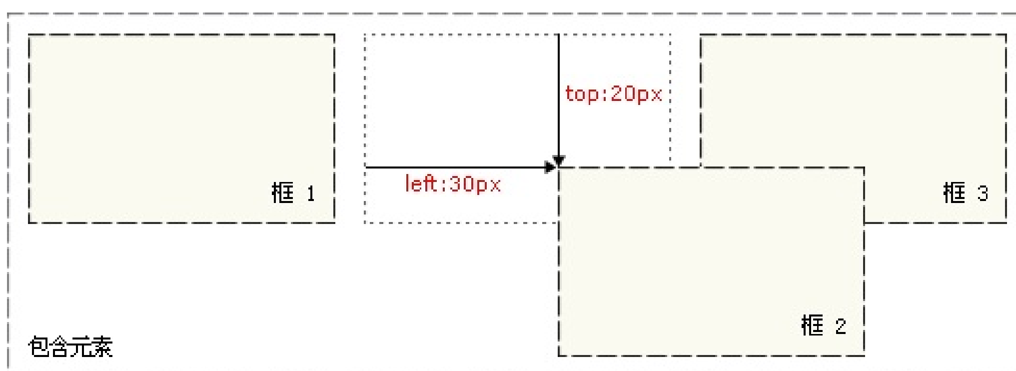
CSS 相对定位

相对定位是一个非常容易掌握的概念。如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。

如果将 top 设置为 20px，那么框将在原位置顶部下面 20 像素的地方。如果 left 设置为 30 像素，那么会在元素左边创建 30 像素的空间，也就是将元素向右移动。

```
#box_relative {
  position: relative;
  left: 30px;
  top: 20px;
}
```

如下图所示：



注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

CSS 相对定位实例

定位：相对定位

CSS 绝对定位

设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位，包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

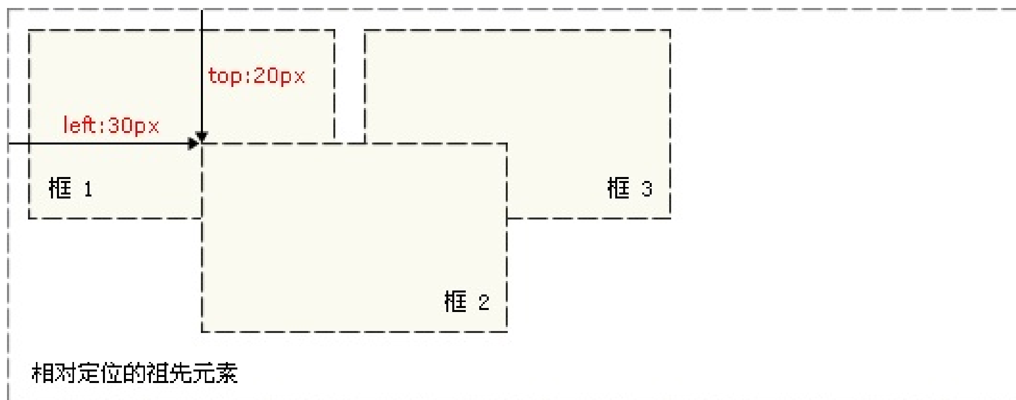
CSS 绝对定位

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其它元素的布局就像绝对定位的元素不存在一样：

```
#box_relative {
  position: absolute;
  left: 30px;
  top: 20px;
}
```

如下图所示：



绝对定位的元素的位置相对于最近的已定位祖先元素，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块。

对于定位的主要问题是要记住每种定位的意义。所以，现在让我们复习一下学过的知识吧：相对定位是“相对于”元素在文档中的初始位置，而绝对定位是“相对于”最近的已定位祖先元素，如果不存在已定位的祖先元素，那么“相对于”最初的包含块。

注释：根据用户代理的不同，最初的包含块可能是画布或 HTML 元素。

提示：因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 z-index 属性来控制这些框的堆放次序。

CSS 绝对定位实例

定位：绝对定位

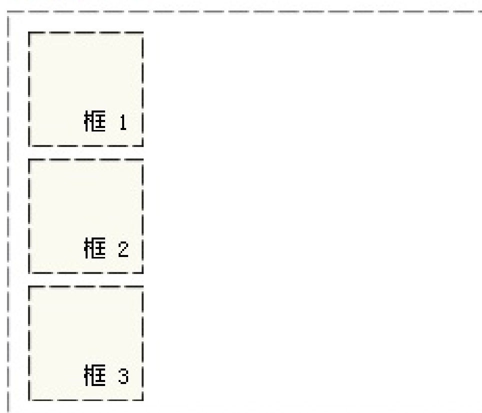
CSS 浮动

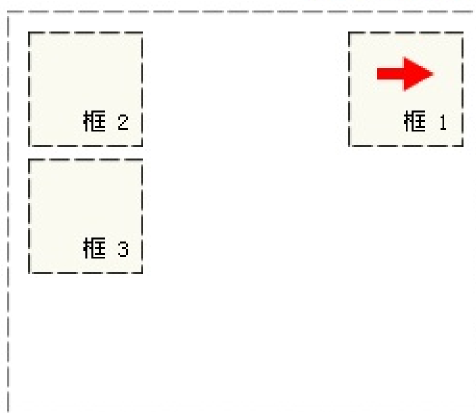
浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

CSS 浮动

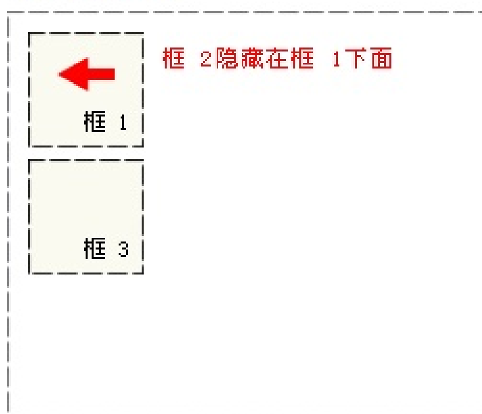
请看下图，当把框 1 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘：

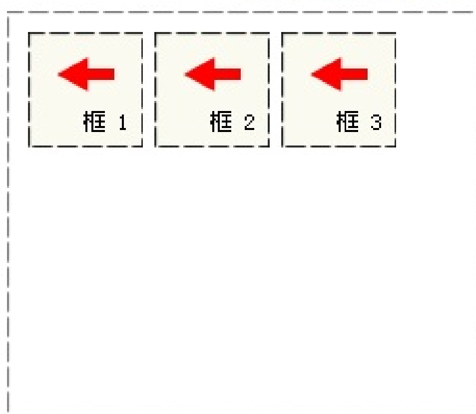




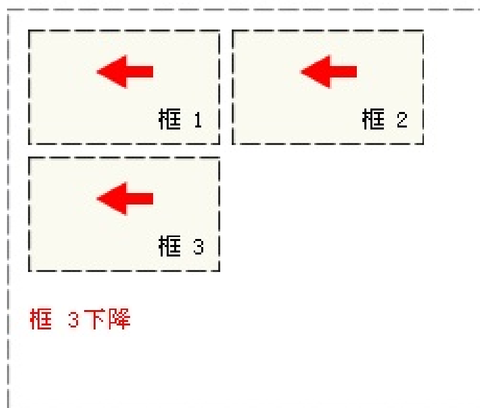
再请看下图，当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失。

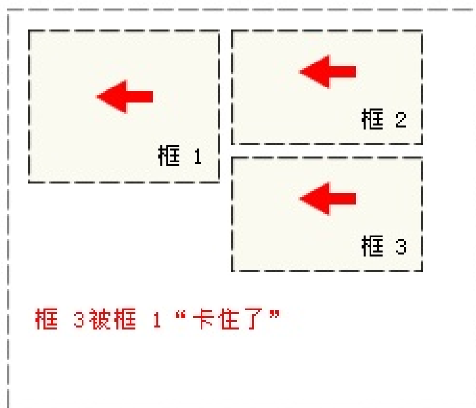
如果把所有三个框都向左移动，那么框 1 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。





如下图所示，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”：





CSS float 属性

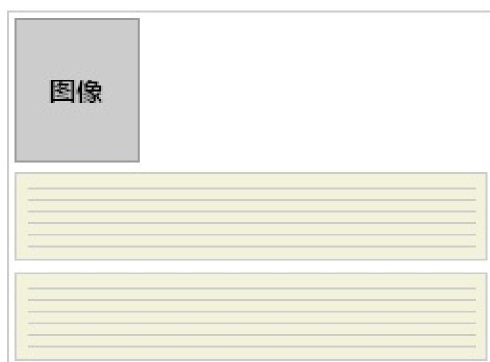
在 CSS 中，我们通过 float 属性实现元素的浮动。

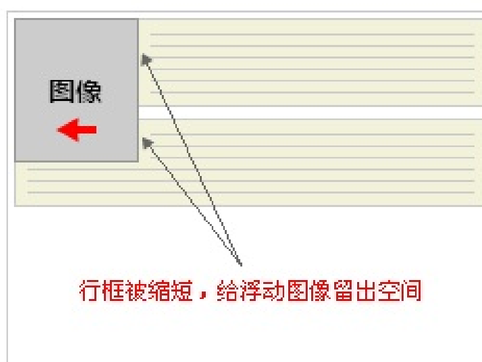
如需更多有关 float 属性的知识，请访问参考手册：CSS float 属性。

行框和清理

浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。

因此，创建浮动框可以使文本围绕图像：





要想阻止行框围绕浮动框，需要对该框应用 clear 属性。clear 属性的值可以是 left、right、both 或 none，它表示框的哪些边不应该挨着浮动框。

为了实现这种效果，在被清理的元素的上外边距上添加足够的空间，使元素的顶边缘垂直下降到浮动框下面：



这是一个有用的工具，它让周围的元素为浮动元素留出空间。

让我们更详细地看看浮动和清理。假设希望让一个图片浮动到文本块的左边，并且希望这幅图片和文本包含在另一个具有背景颜色和边框的元素中。您可能编写下面的代码：

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

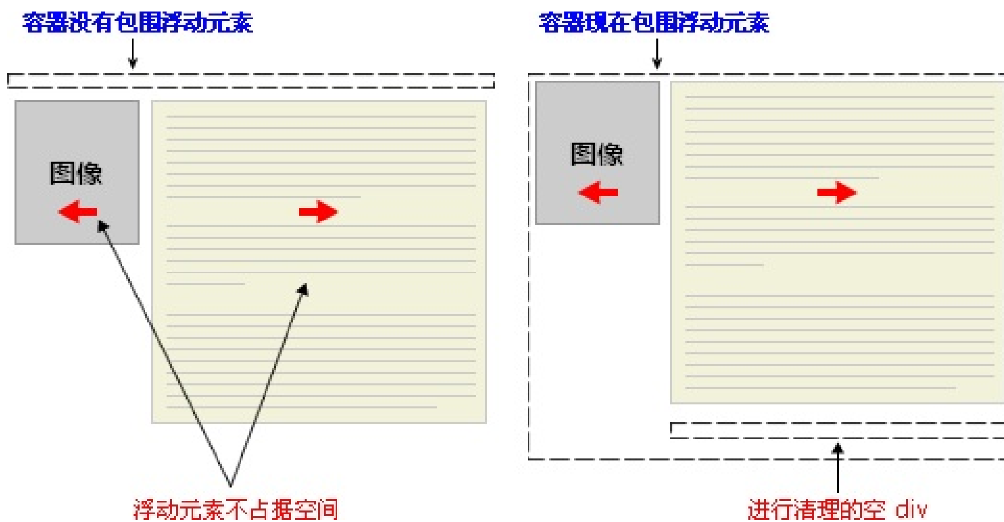
.news img {
  float: left;
  }

.news p {
  float: right;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
</div>
```

这种情况下，出现了一个问题。因为浮动元素脱离了文档流，所以包围图片和文本的 div 不占据空间。

如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用 clear：



不幸的是出现了一个新的问题，由于没有现有的元素可以应用清理，所以我们只能添加一个空元素并且清理它。

```
.news {
  background-color: gray;
  border: solid 1px black;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

.clear {
  clear: both;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
<div class="clear"></div>
</div>
```

这样可以实现我们希望的效果，但是需要添加多余的代码。常常有元素可以应用 clear，但是有时候不得不为了进行布局而添加无意义的标记。

不过我们还有另一种办法，那就是对容器 div 进行浮动：

```
.news {
  background-color: gray;
  border: solid 1px black;
  float: left;
  }

.news img {
  float: left;
  }

.news p {
  float: right;
  }

<div class="news">
<img src="news-pic.jpg" />
<p>some text</p>
</div>
```

这样会得到我们希望的效果。不幸的是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择对布局中的所有东西进行浮动，然后使用适当的有意义的元素（常常是站点的页脚）对这些浮动进行清理。这有助于减少或消除不必要的标记。

事实上，W3School 站点上的所有页面都采用了这种技术，如果您打开我们使用 CSS 文件，您会看到我们对页脚的 div 进行了清理，而页脚上面的三个 div 都向左浮动。

CSS clear 属性

我们刚才详细讨论了 CSS 清理的工作原理和 clear 属性应用方法。如果您希望学习更多有关 clear 属性的知识，请访问参考手册：CSS clear 属性。

浮动和清理 实例

float 属性的简单应用

将带有边框和边界的图像浮动于段落的右侧

带标题的图像浮动于右侧

使段落的首字母浮动于左侧

创建水平菜单

创建无表格的首页

清除元素的侧面

CSS 选择器

CSS 元素选择器

CSS 元素选择器

最常见的 CSS 选择器是元素选择器。换句话说，文档的元素就是最基本的选择器。

如果设置 HTML 的样式，选择器通常将是某个 HTML 元素，比如 p、h1、em、a，甚至可以是 html 本身：

```
html {color:black;}
h1 {color:blue;}
h2 {color:silver;}
```

可以将某个样式从一个元素切换到另一个元素。

假设您决定将上面的段落文本（而不是 h1 元素）设置为灰色。只需要把 h1 选择器改为 p：

```
html {color:black;}
p {color:gray;}
h2 {color:silver;}
```

类型选择器

在 W3C 标准中，元素选择器又称为类型选择器（type selector）。

“类型选择器匹配文档语言元素类型的名称。类型选择器匹配文档树中该元素类型的每一个实例。”

下面的规则匹配文档树中所有 h1 元素：

```
h1 {font-family: sans-serif;}
```

因此，我们也可以为 XML 文档中的元素设置样式：

XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="note.css"?>
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

CSS 文档：

```
note
  {
  font-family:Verdana, Arial;
  margin-left:30px;
  }

to
  {
  font-size:28px;
  display: block;
  }

from
  {
  font-size:28px;
  display: block;
  }

heading
  {
  color: red;
  font-size:60px;
  display: block;
  }

body
  {
  color: blue;
  font-size:35px;
  display: block;
  }
```

查看效果

通过上面的例子，您可以看到，CSS 元素选择器（类型选择器）可以设置 XML 文档中元素的样式。

如果您需要更多关于为 XML 文档添加样式的知识，请访问 w3school 的 XML 教程。

CSS 分组

选择器分组

假设希望 h2 元素和段落都有灰色。为达到这个目的，最容易的做法是使用以下声明：

```
h2, p {color:gray;}
```

将 h2 和 p 选择器放在规则左边，然后用逗号分隔，就定义了一个规则。其右边的样式（color:gray;）将应用到这两个选择器所引用的元素。逗号告诉浏览器，规则中包含两个不同的选择器。如果没有这个逗号，那么规则的含义将完全不同。参见后代选择器。

可以将任意多个选择器分组在一起，对此没有任何限制。

例如，如果您想把很多元素显示为灰色，可以使用类似如下的规则：

```
body, h2, p, table, th, td, pre, strong, em {color:gray;}
```

提示：通过分组，创作者可以将某些类型的样式"压缩"在一起，这样就可以得到更简洁的样式表。

以下的两组规则能得到同样的结果，不过可以很清楚地看出哪一个写起来更容易：

```
/* no grouping */
h1 {color:blue;}
h2 {color:blue;}
h3 {color:blue;}
h4 {color:blue;}
h5 {color:blue;}
h6 {color:blue;}

/* grouping */
h1, h2, h3, h4, h5, h6 {color:blue;}
```

分组提供了一些有意思的选择。例如，下例中的所有规则分组都是等价的，每个组只是展示了对选择器和声明分组的不同方法：

```
/* group 1 */
h1 {color:silver; background:white;}
h2 {color:silver; background:gray;}
h3 {color:white; background:gray;}
h4 {color:silver; background:white;}
b {color:gray; background:white;}

/* group 2 */
h1, h2, h4 {color:silver;}
h2, h3 {background:gray;}
h1, h4, b {background:white;}
h3 {color:white;}
b {color:gray;}

/* group 3 */
h1, h4 {color:silver; background:white;}
h2 {color:silver;}
h3 {color:white;}
h2, h3 {background:gray;}
b {color:gray; background:white;}
```

亲自试一试：

- group 1
- group 2
- group 3

请注意，group 3 中使用了“声明分组”。稍后我们会为您介绍“声明分组”。

通配符选择器

CSS2 引入了一种新的简单选择器 - 通配选择器（universal selector），显示为一个星号（*）。该选择器可以与任何元素匹配，就像是一个通配符。

例如，下面的规则可以使文档中的每个元素都为红色：

```
* {color:red;}
```

这个声明等价于列出了文档中所有元素的一个分组选择器。利用通配选择器，只需敲一次键（仅一个星号）就能使文档中所有元素的 color 属性值指定为 red。

声明分组

我们既可以对选择器进行分组，也可以对声明分组。

假设您希望所有 h1 元素都有红色背景，并使用 28 像素高的 Verdana 字体显示为蓝色文本，可以写以下样式：

```
h1 {font: 28px Verdana;}
h1 {color: blue;}
h1 {background: red;}
```

但是上面这种做法的效率并不高。尤其是当我们为一个有多个样式的元素创建这样一个列表时会很麻烦。相反，我们可以将声明分组在一起：

```
h1 {font: 28px Verdana; color: white; background: black;}
```

这与前面的 3 行样式表的效果完全相同。

注意，对声明分组，一定要在各个声明的最后使用分号，这很重要。浏览器会忽略样式表中的空白符。只要加了分号，就可以毫无顾忌地采用以下格式建立样式：

```
h1 {
  font: 28px Verdana;
  color: blue;
  background: red;
  }
```

怎么样，上面这种写法的可读性是不是更强。

不过，如果忽略了第二个分号，用户代理就会把这个样式表解释如下：

```
h1 {
  font: 28px Verdana;
  color: blue background: red;
  }
```

因为 background 对 color 来说不是一个合法值，而且由于只能为 color 指定一个关键字，所以用户代理会完全忽略这个 color 声明（包括 background: black 部分）。这样 h1 标题只会显示为蓝色，而没有红色背景，不过更有可能根本得不到蓝色的 h1。相反，这些标题只会显示为默认颜色（通常是黑色），而且根本没有背景色。font: 28px Verdana 声明仍能正常发挥作用，因为它确实正确地以一个分号结尾。

与选择器分组一样，声明分组也是一种便利的方法，可以缩短样式表，使之更清晰，也更易维护。

提示：在规则的最后一个声明后也加上分号是一个好习惯。在向规则增加另一个声明时，就不必担心忘记再插入一个分号。

结合选择器和声明的分组

我们可以在一个规则中结合选择器分组和声明分组，就可以使用很少的语句定义相对复杂的样式。

下面的规则为所有标题指定了一种复杂的样式：

```
h1, h2, h3, h4, h5, h6 {
  color:gray;
  background: white;
  padding: 10px;
  border: 1px solid black;
  font-family: Verdana;
  }
```

上面这条规则将所有标题的样式定义为带有白色背景的灰色文本，其内边距是 10 像素，并带有 1 像素的实心边框，文本字体是 Verdana。

CSS 类选择器详解

类选择器允许以一种独立于文档元素的方式来指定样式。

CSS 类选择器

类选择器允许以一种独立于文档元素的方式来指定样式。

该选择器可以单独使用，也可以与其他元素结合使用。

提示：只有适当地标记文档后，才能使用这些选择器，所以使用这两种选择器通常需要先做一些构想和计划。

要应用样式而不考虑具体设计的元素，最常用的方法就是使用类选择器。

修改 HTML 代码

在使用类选择器之前，需要修改具体的文档标记，以便类选择器正常工作。

为了将类选择器的样式与元素关联，必须将 class 指定为一个适当的值。请看下面的 HTML 代码：

```
<h1 class="important">
This heading is very important.
</h1>

<p class="important">
This paragraph is very important.
</p>
```

在上面的代码中，两个元素的 class 都指定为 important：第一个标题（ h1 元素），第二个段落（p 元素）。

语法

然后我们使用以下语法向这些归类的元素应用样式，即类名前有一个点号（.），然后结合通配选择器：

```
*.important {color:red;}
```

如果您只想选择所有类名相同的元素，可以在类选择器中忽略通配选择器，这没有任何不好的影响：

```
.important {color:red;}
```

结合元素选择器

类选择器可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本：

```
p.important {color:red;}
```

选择器现在会匹配 class 属性包含 important 的所有 p 元素，但是其他任何类型的元素都不匹配，不论是否有此 class 属性。选择器 p.important 解释为："其 class 属性值为 important 的所有段落"。 因为 h1 元素不是段落，这个规则的选择器与之不匹配，因此 h1 元素不会变成红色文本。

如果你确实希望为 h1 元素指定不同的样式，可以使用选择器 h1.important：

```
p.important {color:red;}
h1.important {color:blue;}
```

CSS 多类选择器

在上一节中，我们处理了 class 值中包含一个词的情况。在 HTML 中，一个 class 值中可能包含一个词列表，各个词之间用空格分隔。例如，如果希望将一个特定的元素同时标记为重要（important）和警告（warning），就可以写作：

```
<p class="important warning">
This paragraph is a very important warning.
</p>
```

这两个词的顺序无关紧要，写成 warning important 也可以。

我们假设 class 为 important 的所有元素都是粗体，而 class 为 warning 的所有元素为斜体，class 中同时包含 important 和 warning 的所有元素还有一个银色的背景。就可以写作：

```
.important {font-weight:bold;}
.warning {font-style:italic;}
.important.warning {background:silver;}
```

通过把两个类选择器链接在一起，仅可以选择同时包含这些类名的元素（类名的顺序不限）。

如果一个多类选择器包含类名列表中没有的一个类名，匹配就会失败。请看下面的规则：

```
.important.urgent {background:silver;}
```

不出所料，这个选择器将只匹配 class 属性中包含词 important 和 urgent 的 p 元素。因此，如果一个 p 元素的 class 属性中只有词 important 和 warning，将不能匹配。不过，它能匹配以下元素：

```
<p class="important urgent warning">
This paragraph is a very important and urgent warning.
</p>
```

重要事项：在 IE7 之前的版本中，不同平台的 Internet Explorer 都不能正确地处理多类选择器。

CSS ID 选择器详解

ID 选择器允许以一种独立于文档元素的方式来指定样式。

CSS ID 选择器

在某些方面，ID 选择器类似于类选择器，不过也有一些重要差别。

语法

首先，ID 选择器前面有一个 # 号 - 也称为棋盘号或井号。

请看下面的规则：

```
*#intro {font-weight:bold;}
```

与类选择器一样，ID 选择器中可以忽略通配选择器。前面的例子也可以写作：

```
#intro {font-weight:bold;}
```

这个选择器的效果将是一样的。

第二个区别是 ID 选择器不引用 class 属性的值，毫无疑问，它要引用 id 属性中的值。

以下是一个实际 ID 选择器的例子：

```
<p id="intro">This is a paragraph of introduction.</p>
```

类选择器还是 ID 选择器？

在类选择器这一章中我们曾讲解过，可以为任意多个元素指定类。前一章中类名 important 被应用到 p 和 h1 元素，而且它还可以应用到更多元素。

区别 1：只能在文档中使用一次

与类不同，在一个 HTML 文档中，ID 选择器会使用一次，而且仅一次。

区别 2：不能使用 ID 词列表

不同于类选择器，ID 选择器不能结合使用，因为 ID 属性不允许有以空格分隔的词列表。

区别 3：ID 能包含更多含义

类似于类，可以独立于元素来选择 ID。有些情况下，您知道文档中会出现某个特定 ID 值，但是并不知道它会出现在哪个元素上，所以您想声明独立的 ID 选择器。例如，您可能知道在一个给定的文档中会有一个 ID 值为 mostImportant 的元素。您不知道这个最重要的东西是一个段落、一个短语、一个列表项还是一个小节标题。您只知道每个文档都会有这么一个最重要的内容，它可能在任何元素中，而且只能出现一个。在这种情况下，可以编写如下规则：

```
#mostImportant {color:red; background:yellow;}
```

这个规则会与以下各个元素匹配（这些元素不能在同一个文档中同时出现，因为它们都有相同的 ID 值）：

```
<h1 id="mostImportant">This is important!</h1>
<em id="mostImportant">This is important!</em>
<ul id="mostImportant">This is important!</ul>
```

亲自试一试：

- 为 id 为 mostImportant 的 h1 元素设定样式
- 为 id 为 mostImportant 的 em 元素设定样式
- 为 id 为 mostImportant 的 ul 元素设定样式

区分大小写

请注意，类选择器和 ID 选择器可能是区分大小写的。这取决于文档的语言。HTML 和 XHTML 将类和 ID 值定义为区分大小写，所以类和 ID 值的大小写必须与文档中的相应值匹配。

因此，对于以下的 CSS 和 HTML，元素不会变成粗体：

```
#intro {font-weight:bold;}

<p id="Intro">This is a paragraph of introduction.</p>
```

由于字母 i 的大小写不同，所以选择器不会匹配上面的元素。

CSS 属性选择器详解

CSS 2 引入了属性选择器。

属性选择器可以根据元素的属性及属性值来选择元素。

简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

例子 1

如果您希望把包含标题（title）的所有元素变为红色，可以写作：

```
*[title] {color:red;}
```

例子 2

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

```
a[href] {color:red;}
```

例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

```
a[href][title] {color:red;}
```

例子 4

可以采用一些创造性的方法使用这个特性。

例如，可以对所有带有 alt 属性的图像应用样式，从而突出显示这些有效的图像：

```
img[alt] {border: 5px solid red;}
```

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

例子 5：为 XML 文档使用属性选择器

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

```
planet[moons] {color:red;}
```

这会让以下标记片段中的第二个和第三个元素的文本显示为红色，但第一个元素的文本不是红色：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

根据具体属性值选择

除了选择拥有某些属性的元素，还可以进一步缩小选择范围，只选择有特定属性值的元素。

例子 1

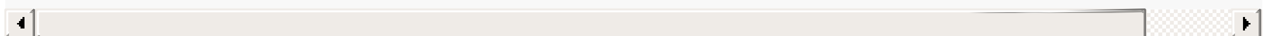
例如，假设希望将指向 Web 服务器上某个指定文档的超链接变成红色，可以这样写：

```
a[href="http://www.w3school.com.cn/about_us.asp"] {color: red;}
```

例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

```
a[href="http://www.w3school.com.cn/"][title="W3School"] {color: rec
```



这会把以下标记中的第一个超链接的文本变为红色，但是第二个或第三个链接不受影响：

```
<a href="http://www.w3school.com.cn/" title="W3School">W3School</a>
<a href="http://www.w3school.com.cn/css/" title="CSS">CSS</a>
<a href="http://www.w3school.com.cn/html/" title="HTML">HTML</a>
```

例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

下面我们再回到行星那个例子中。假设只希望选择 moons 属性值为 1 的那些 planet 元素：

```
planet[moons="1"] {color: red;}
```

上面的代码会把以下标记中的第二个元素变成红色，但第一个和第三个元素不受影响：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

属性与属性值必须完全匹配

请注意，这种格式要求必须与属性值完全匹配。

如果属性值包含用空格分隔的值列表，匹配就可能出问题。

请考虑一下的标记片段：

```
<p class="important warning">This paragraph is a very important wa
```

如果写成 p[class="important"]，那么这个规则不能匹配示例标记。

要根据具体属性值来选择该元素，必须这样写：

```
p[class="important warning"] {color: red;}
```

根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

```
p[class~="important"] {color: red;}
```

如果忽略了波浪号，则说明需要完成完全值匹配。

部分值属性选择器与点号类名记法的区别

该选择器等价于我们在类选择器中讨论过的点号类名记法。

也就是说，p.important 和 p[class="important"] 应用到 HTML 文档时是等价的。

那么，为什么还要有 "~=" 属性选择器呢？因为它能用于任何属性，而不只是 class。

例如，可以有一个包含大量图像的文档，其中只有一部分是图片。对此，可以使用一个基于 title 文档的部分属性选择器，只选择这些图片：

```
img[title~="Figure"] {border: 1px solid gray;}
```

这个规则会选择 title 文本包含 "Figure" 的所有图像。没有 title 属性或者 title 属性中不包含 "Figure" 的图像都不会匹配。

子串匹配属性选择器

下面为您介绍一个更高级的选择器模块，它是 CSS2 完成之后发布的，其中包含了更多的部分值属性选择器。按照规范的说法，应该称之为“子串匹配属性选择器”。

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

类型	描述
[abc^="def"]	选择 abc 属性值以 "def" 开头的所有元素
[abc$="def"]	选择 abc 属性值以 "def" 结尾的所有元素
[abc*="def"]	选择 abc 属性值中包含子串 "def" 的所有元素

可以想到，这些选择有很多用途。

举例来说，如果希望对指向 W3School 的所有链接应用样式，不必为所有这些链接指定 class，再根据这个类编写样式，而只需编写以下规则：

```
a[href*="w3school.com.cn"] {color: red;}
```

提示：任何属性都可以使用这些选择器。

特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

```
*[lang|="en"] {color: red;}
```

上面这个规则会选择 lang 属性等于 en 或以 en- 开头的所有元素。因此，以下示例标记中的前三个元素将被选中，而不会选择后两个元素：

```
<p lang="en">Hello!</p>
<p lang="en-us">Greetings!</p>
<p lang="en-au">G'day!</p>
<p lang="fr">Bonjour!</p>
<p lang="cy-en">Jrooana!</p>
```

一般来说，[att|="val"] 可以用于任何属性及其值。

假设一个 HTML 文档中有一系列图片，其中每个图片的文件名都形如 *figure-1.jpg* 和 *figure-2.jpg*。就可以使用以下选择器匹配所有这些图像：

```
img[src|="figure"] {border: 1px solid gray;}
```

当然，这种属性选择器最常见的用途还是匹配语言值。

CSS 选择器参考手册

选择器	描述
[attribute]	用于选取带有指定属性的元素。
[attribute=value]	用于选取带有指定属性和值的元素。
[attribute~=value]	用于选取属性值中包含指定词汇的元素。
[attribute|=value]	用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

CSS 后代选择器

后代选择器（**descendant selector**）又称为包含选择器。

后代选择器可以选择作为某元素后代的元素。

根据上下文选择元素

我们可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

```
h1 em {color:red;}
```

上面这个规则会把作为 h1 元素后代的 em 元素的文本变为 红色。其他 em 文本（如段落或块引用中的 em）则不会被这个规则选中：

```
<h1>This is a <em>important</em> heading</h1>
<p>This is a <em>important</em> paragraph.</p>
```

当然，您也可以在 h1 中找到的每个 em 元素上放一个 class 属性，但是显然，后代选择器的效率更高。

语法解释

在后代选择器中，规则左边的选择器一端包括两个或多个用空格分隔的选择器。选择器之间的空格是一种结合符（combinator）。每个空格结合符可以解释为“... 在 ... 找到”、“... 作为 ... 的一部分”、“... 作为 ... 的后代”，但是要求必须从右向左读选择器。

因此，h1 em 选择器可以解释为 “作为 h1 元素后代的任何 em 元素”。如果要从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

```
div.sidebar {background:blue;}
div.maincontent {background:white;}
div.sidebar a:link {color:white;}
div.maincontent a:link {color:blue;}
```

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

```
<ul>
  <li>List item 1
    <ol>
      <li>List item 1-1</li>
      <li>List item 1-2</li>
      <li>List item 1-3
        <ol>
          <li>List item 1-3-1</li>
          <li>List item <em>1-3-2</em></li>
          <li>List item 1-3-3</li>
        </ol>
      </li>
      <li>List item 1-4</li>
    </ol>
  </li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

CSS 属性选择器详解

CSS 2 引入了属性选择器。

属性选择器可以根据元素的属性及属性值来选择元素。

简单属性选择

如果希望选择有某个属性的元素，而不论属性值是什么，可以使用简单属性选择器。

例子 1

如果您希望把包含标题（title）的所有元素变为红色，可以写作：

```
*[title] {color:red;}
```

例子 2

与上面类似，可以只对有 href 属性的锚（a 元素）应用样式：

```
a[href] {color:red;}
```

例子 3

还可以根据多个属性进行选择，只需将属性选择器链接在一起即可。

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

```
a[href][title] {color:red;}
```

例子 4

可以采用一些创造性的方法使用这个特性。

例如，可以对所有带有 alt 属性的图像应用样式，从而突出显示这些有效的图像：

```
img[alt] {border: 5px solid red;}
```

提示：上面这个特例更适合用来诊断而不是设计，即用来确定图像是否确实有效。

例子 5：为 XML 文档使用属性选择器

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

```
planet[moons] {color:red;}
```

这会让以下标记片段中的第二个和第三个元素的文本显示为红色，但第一个元素的文本不是红色：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

根据具体属性值选择

除了选择拥有某些属性的元素，还可以进一步缩小选择范围，只选择有特定属性值的元素。

例子 1

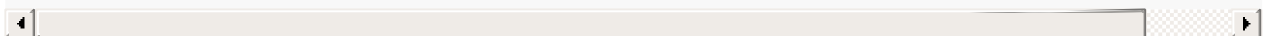
例如，假设希望将指向 Web 服务器上某个指定文档的超链接变成红色，可以这样写：

```
a[href="http://www.w3school.com.cn/about_us.asp"] {color: red;}
```

例子 2

与简单属性选择器类似，可以把多个属性-值选择器链接在一起来选择一个文档。

```
a[href="http://www.w3school.com.cn/"][title="W3School"] {color: rec
```



这会把以下标记中的第一个超链接的文本变为红色，但是第二个或第三个链接不受影响：

```
<a href="http://www.w3school.com.cn/" title="W3School">W3School</a>
<a href="http://www.w3school.com.cn/css/" title="CSS">CSS</a>
<a href="http://www.w3school.com.cn/html/" title="HTML">HTML</a>
```

例子 3

同样地，XML 语言也可以利用这种方法来设置样式。

下面我们再回到行星那个例子中。假设只希望选择 moons 属性值为 1 的那些 planet 元素：

```
planet[moons="1"] {color: red;}
```

上面的代码会把以下标记中的第二个元素变成红色，但第一个和第三个元素不受影响：

```
<planet>Venus</planet>
<planet moons="1">Earth</planet>
<planet moons="2">Mars</planet>
```

查看效果

属性与属性值必须完全匹配

请注意，这种格式要求必须与属性值完全匹配。

如果属性值包含用空格分隔的值列表，匹配就可能出问题。

请考虑一下的标记片段：

```
<p class="important warning">This paragraph is a very important wa
```

如果写成 p[class="important"]，那么这个规则不能匹配示例标记。

要根据具体属性值来选择该元素，必须这样写：

```
p[class="important warning"] {color: red;}
```

根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

```
p[class~="important"] {color: red;}
```

如果忽略了波浪号，则说明需要完成完全值匹配。

部分值属性选择器与点号类名记法的区别

该选择器等价于我们在类选择器中讨论过的点号类名记法。

也就是说，p.important 和 p[class="important"] 应用到 HTML 文档时是等价的。

那么，为什么还要有 "~=" 属性选择器呢？因为它能用于任何属性，而不只是 class。

例如，可以有一个包含大量图像的文档，其中只有一部分是图片。对此，可以使用一个基于 title 文档的部分属性选择器，只选择这些图片：

```
img[title~="Figure"] {border: 1px solid gray;}
```

这个规则会选择 title 文本包含 "Figure" 的所有图像。没有 title 属性或者 title 属性中不包含 "Figure" 的图像都不会匹配。

子串匹配属性选择器

下面为您介绍一个更高级的选择器模块，它是 CSS2 完成之后发布的，其中包含了更多的部分值属性选择器。按照规范的说法，应该称之为“子串匹配属性选择器”。

很多现代浏览器都支持这些选择器，包括 IE7。

下表是对这些选择器的简单总结：

类型	描述
[abc^="def"]	选择 abc 属性值以 "def" 开头的所有元素
[abc$="def"]	选择 abc 属性值以 "def" 结尾的所有元素
[abc*="def"]	选择 abc 属性值中包含子串 "def" 的所有元素

可以想到，这些选择有很多用途。

举例来说，如果希望对指向 W3School 的所有链接应用样式，不必为所有这些链接指定 class，再根据这个类编写样式，而只需编写以下规则：

```
a[href*="w3school.com.cn"] {color: red;}
```

提示：任何属性都可以使用这些选择器。

特定属性选择类型

最后为您介绍特定属性选择器。请看下面的例子：

```
*[lang|="en"] {color: red;}
```

上面这个规则会选择 lang 属性等于 en 或以 en- 开头的所有元素。因此，以下示例标记中的前三个元素将被选中，而不会选择后两个元素：

```
<p lang="en">Hello!</p>
<p lang="en-us">Greetings!</p>
<p lang="en-au">G'day!</p>
<p lang="fr">Bonjour!</p>
<p lang="cy-en">Jrooana!</p>
```

一般来说，[att|="val"] 可以用于任何属性及其值。

假设一个 HTML 文档中有一系列图片，其中每个图片的文件名都形如 *figure-1.jpg* 和 *figure-2.jpg*。就可以使用以下选择器匹配所有这些图像：

```
img[src|="figure"] {border: 1px solid gray;}
```

当然，这种属性选择器最常见的用途还是匹配语言值。

CSS 选择器参考手册

选择器	描述
[attribute]	用于选取带有指定属性的元素。
[attribute=value]	用于选取带有指定属性和值的元素。
[attribute~=value]	用于选取属性值中包含指定词汇的元素。
[attribute|=value]	用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

CSS 后代选择器

后代选择器（**descendant selector**）又称为包含选择器。

后代选择器可以选择作为某元素后代的元素。

根据上下文选择元素

我们可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。

举例来说，如果您希望只对 h1 元素中的 em 元素应用样式，可以这样写：

```
h1 em {color:red;}
```

上面这个规则会把作为 h1 元素后代的 em 元素的文本变为 红色。其他 em 文本（如段落或块引用中的 em）则不会被这个规则选中：

```
<h1>This is a <em>important</em> heading</h1>
<p>This is a <em>important</em> paragraph.</p>
```

当然，您也可以在 h1 中找到的每个 em 元素上放一个 class 属性，但是显然，后代选择器的效率更高。

语法解释

在后代选择器中，规则左边的选择器一端包括两个或多个用空格分隔的选择器。选择器之间的空格是一种结合符（combinator）。每个空格结合符可以解释为“... 在 ... 找到”、“... 作为 ... 的一部分”、“... 作为 ... 的后代”，但是要求必须从右向左读选择器。

因此，h1 em 选择器可以解释为 “作为 h1 元素后代的任何 em 元素”。如果要从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

```
div.sidebar {background:blue;}
div.maincontent {background:white;}
div.sidebar a:link {color:white;}
div.maincontent a:link {color:blue;}
```

有关后代选择器有一个易被忽视的方面，即两个元素之间的层次间隔可以是无限的。

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

```
<ul>
  <li>List item 1
    <ol>
      <li>List item 1-1</li>
      <li>List item 1-2</li>
      <li>List item 1-3
        <ol>
          <li>List item 1-3-1</li>
          <li>List item <em>1-3-2</em></li>
          <li>List item 1-3-3</li>
        </ol>
      </li>
      <li>List item 1-4</li>
    </ol>
  </li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

CSS 子元素选择器

与后代选择器相比，子元素选择器（**Child selectors**）只能选择作为某元素子元素的元素。

选择子元素

如果您不希望选择任意的后代元素，而是希望缩小范围，只选择某个元素的子元素，请使用子元素选择器（Child selector）。

例如，如果您希望选择只作为 h1 元素子元素的 strong 元素，可以这样写：

```
h1 > strong {color:red;}
```

这个规则会把第一个 h1 下面的两个 strong 元素变为红色，但是第二个 h1 中的 strong 不受影响：

```
<h1>This is <strong>very</strong> <strong>very</strong> important.<
<h1>This is <em>really <strong>very</strong></em> important.</h1>
```

语法解释

您应该已经注意到了，子选择器使用了大于号（子结合符）。

子结合符两边可以有空白符，这是可选的。因此，以下写法都没有问题：

```
h1 > strong
h1> strong
h1 >strong
h1>strong
```

如果从右向左读，选择器 h1 > strong 可以解释为“选择作为 h1 元素子元素的所有 strong 元素”。

结合后代选择器和子选择器

请看下面这个选择器：

```
table.company td > p
```

上面的选择器会选择作为 td 元素子元素的所有 p 元素，这个 td 元素本身从 table 元素继承，该 table 元素有一个包含 company 的 class 属性。

CSS 相邻兄弟选择器

相邻兄弟选择器（**Adjacent sibling selector**）可选择紧接在另一元素后的元素，且二者有相同父元素。

选择相邻兄弟

如果需要选择紧接在另一个元素后的元素，而且二者有相同的父元素，可以使用相邻兄弟选择器（Adjacent sibling selector）。

例如，如果要增加紧接在 h1 元素后出现的段落的上边距，可以这样写：

```
h1 + p {margin-top:50px;}
```

这个选择器读作："选择紧接在 h1 元素后出现的段落，h1 和 p 元素拥有共同的父元素"。

语法解释

相邻兄弟选择器使用了加号（+），即相邻兄弟结合符（Adjacent sibling combinator）。

注释：与子结合符一样，相邻兄弟结合符旁边可以有空白符。

请看下面这个文档树片段：

```
<div>
  <ul>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
  </ul>
  <ol>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
  </ol>
</div>
```

在上面的片段中，div 元素中包含两个列表：一个无序列表，一个有序列表，每个列表都包含三个列表项。这两个列表是相邻兄弟，列表项本身也是相邻兄弟。不过，第一个列表中的列表项与第二个列表中的列表项不是相邻兄弟，因为这两组列表项不属于同一父元素（最多只能算堂兄弟）。

请记住，用一个结合符只能选择两个相邻兄弟中的第二个元素。请看下面的选择器：

```
li + li {font-weight:bold;}
```

上面这个选择器只会把列表中的第二个和第三个列表项变为粗体。第一个列表项不受影响。

结合其他选择器

相邻兄弟结合符还可以结合其他结合符：

```
html > body table + ul {margin-top:20px;}
```

这个选择器解释为：选择紧接在 table 元素后出现的所有兄弟 ul 元素，该 table 元素包含在一个 body 元素中，body 元素本身是 html 元素的子元素。

CSS 伪类 (Pseudo-classes)

CSS 伪类用于向某些选择器添加特殊的效果。

CSS 伪类 (Pseudo-classes)实例：

超链接

```
&lt;html&gt;
&lt;head&gt;

&lt;style type="text/css"&gt;
a:link {color: #FF0000}
a:visited {color: #00FF00}
a:hover {color: #FF00FF}
a:active {color: #0000FF}
&lt;/style&gt;

&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;&lt;b&gt;&lt;a href="/index.html" target="_blank"&gt;这是一
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 CSS 定义中, a:hover 必须位于 a:lin
&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 CSS 定义中, a:active 必须位于 a:ho

&lt;/body&gt;
&lt;/html&gt;
```

超链接 2

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
a.one:link {color: #ff0000}
a.one:visited {color: #0000ff}
a.one:hover {color: #ffcc00}

a.two:link {color: #ff0000}
a.two:visited {color: #0000ff}
a.two:hover {font-size: 150%}

a.three:link {color: #ff0000}
a.three:visited {color: #0000ff}
a.three:hover {background: #66ff66}

a.four:link {color: #ff0000}
a.four:visited {color: #0000ff}
a.four:hover {font-family: monospace}

a.five:link {color: #ff0000; text-decoration: none}
a.five:visited {color: #0000ff; text-decoration: none}
a.five:hover {text-decoration: underline}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;请把鼠标移动到这些链接上，以查看效果：&lt;/p&gt;

&lt;p&gt;&lt;b&gt;&lt;a class="one" href="/index.html" target="_bla
&lt;p&gt;&lt;b&gt;&lt;a class="two" href="/index.html" target="_bla
&lt;p&gt;&lt;b&gt;&lt;a class="three" href="/index.html" target="_b
&lt;p&gt;&lt;b&gt;&lt;a class="four" href="/index.html" target="_bl
&lt;p&gt;&lt;b&gt;&lt;a class="five" href="/index.html" target="_bl
&lt;/body&gt;

&lt;/html&gt;
```

超链接 - :focus 的使用

```
&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
input:focus
{
background-color:yellow;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;form action="form_action.asp" method="get"&gt;
First name: &lt;input type="text" name="fname" /&gt;&lt;br /&gt;
Last name: &lt;input type="text" name="lname" /&gt;&lt;br /&gt;
&lt;input type="submit" value="Submit" /&gt;
&lt;/form&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;如果已规定 !DOCTYPE，那么 Internet E

&lt;/body&gt;
&lt;/html&gt;
```

:first-child（首个子对象）

```
&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" '
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p:first-child {font-weight: bold;}
li:first-child {text-transform:uppercase;}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;div&gt;
&lt;p&gt;These are the necessary steps:&lt;/p&gt;
&lt;ul&gt;
&lt;li&gt;Intert Key&lt;/li&gt;
&lt;li&gt;Turn key &lt;strong&gt;clockwise&lt;/strong&gt;&lt;/li&gt
&lt;li&gt;Push accelerator&lt;/li&gt;
&lt;/ul&gt;
&lt;p&gt;Do &lt;em&gt;not&lt;/em&gt; push the brake at the same tir
&lt;/div&gt;

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;必须声明 DOCTYPE，这样 :first-child
&lt;/body&gt;

&lt;/html&gt;
```

:lang（语言）

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
q:lang(no)
{
quotes: "~" "~"
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;:lang 伪类允许您为不同的语言定义特殊的规则。在下面的例子中，在下面的

&lt;p&gt;一些文本 &lt;q lang="no"&gt;段落中的引用&lt;/q&gt; 一些文本。&l
&lt;/body&gt;

&lt;/html&gt;
```

语法

伪类的语法：

```
selector : pseudo-class {property: value}
```

CSS 类也可与伪类搭配使用。

```
selector.class : pseudo-class {property: value}
```

锚伪类

在支持 CSS 的浏览器中，链接的不同状态都可以不同的方式显示，这些状态包括：活动状态，已被访问状态，未被访问状态，和鼠标悬停状态。

```
a:link {color: #FF0000}        /* 未访问的链接 */
a:visited {color: #00FF00}    /* 已访问的链接 */
a:hover {color: #FF00FF}    /* 鼠标移动到链接上 */
a:active {color: #0000FF}    /* 选定的链接 */
```

提示：在 CSS 定义中，a:hover 必须被置于 a:link 和 a:visited 之后，才是有效的。

提示：在 CSS 定义中，a:active 必须被置于 a:hover 之后，才是有效的。

提示：伪类名称对大小写不敏感。

伪类与 CSS 类

伪类可以与 CSS 类配合使用：

```
a.red : visited {color: #FF0000}

<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

假如上面的例子中的链接被访问过，那么它将显示为红色。

CSS2 - :first-child 伪类

您可以使用 :first-child 伪类来选择元素的第一个子元素。这个特定伪类很容易遭到误解，所以有必要举例来说明。考虑以下标记：

```
<div>
<p>These are the necessary steps:</p>
<ul>
<li>Intert Key</li>
<li>Turn key <strong>clockwise</strong></li>
<li>Push accelerator</li>
</ul>
<p>Do <em>not</em> push the brake at the same time as the accelerat
</div>
```

在上面的例子中，作为第一个元素的元素包括第一个 p、第一个 li 和 strong 和 em 元素。

给定以下规则：

```
p:first-child {font-weight: bold;}
li:first-child {text-transform:uppercase;}
```

第一个规则将作为某元素第一个子元素的所有 p 元素设置为粗体。第二个规则将作为某个元素（在 HTML 中，这肯定是 ol 或 ul 元素）第一个子元素的所有 li 元素变成大写。

请访问该链接，来查看这个 :first-child 实例的效果。

提示：最常见的错误是认为 p:first-child 之类的选择器会选择 p 元素的第一个子元素。

注释：必须声明 <!DOCTYPE>，这样 :first-child 才能在 IE 中生效。

为了使您更透彻地理解 :first-child 伪类，我们另外提供了 3 个例子：

例子 1 - 匹配第一个 <p> 元素

在下面的例子中，选择器匹配作为任何元素的第一个子元素的 p 元素：

```
<html>
<head>
<style type="text/css">
p:first-child {
  color: red;
  }
</style>
</head>

<body>
<p>some text</p>
<p>some text</p>
</body>
</html>
```

例子 2 - 匹配所有 <p> 元素中的第一个 <i> 元素

在下面的例子中，选择器匹配所有 <p> 元素中的第一个 <i> 元素：

```
<html>
<head>
<style type="text/css">
p > i:first-child {
  font-weight:bold;
  }
</style>
</head>

<body>
<p>some <i>text</i>. some <i>text</i>.</p>
<p>some <i>text</i>. some <i>text</i>.</p>
</body>
</html>
```

例子 3 - 匹配所有作为第一个子元素的 <p> 元素中的所有 <i> 元素

在下面的例子中，选择器匹配所有作为元素的第一个子元素的 <p> 元素中的所有 <i> 元素：

```
<html>
<head>
<style type="text/css">
p:first-child i {
  color:blue;
  }
</style>
</head>

<body>
<p>some <i>text</i>. some <i>text</i>.</p>
<p>some <i>text</i>. some <i>text</i>.</p>
</body>
</html>
```

CSS2 - :lang 伪类

:lang 伪类使你有能力为不同的语言定义特殊的规则。在下面的例子中，:lang 类为属性值为 no 的 q 元素定义引号的类型：

```
<html>
<head>

<style type="text/css">
q:lang(no)
   {
   quotes: "~" "~"
   }
</style>

</head>

<body>
<p>文字<q lang="no">段落中的引用的文字</q>文字</p>
</body></html>
```

伪类

W3C："W3C" 列指示出该属性在哪个 CSS 版本中定义（CSS1 还是 CSS2）。

属性	描述	CSS
:active	向被激活的元素添加样式。	1
:focus	向拥有键盘输入焦点的元素添加样式。	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式。	1
:link	向未被访问的链接添加样式。	1
:visited	向已被访问的链接添加样式。	1
:first-child	向元素的第一个子元素添加样式。	2
:lang	向带有指定 lang 属性的元素添加样式。	2

CSS 伪元素 (Pseudo-elements)

CSS 伪元素用于向某些选择器设置特殊效果。

语法

伪元素的语法：

```
selector:pseudo-element {property:value;}
```

CSS 类也可以与伪元素配合使用：

```
selector.class:pseudo-element {property:value;}
```

:first-line 伪元素

"first-line" 伪元素用于向文本的首行设置特殊样式。

在下面的例子中，浏览器会根据 "first-line" 伪元素中的样式对 p 元素的第一行文本进行格式化：

实例

```
p:first-line
  {
  color:#ff0000;
  font-variant:small-caps;
  }
```

注释："first-line" 伪元素只能用于块级元素。

注释：下面的属性可应用于 "first-line" 伪元素：

- font
- color
- background
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform

- line-height
- clear

:first-letter 伪元素

"first-letter" 伪元素用于向文本的首字母设置特殊样式：

```
p:first-letter
  {
  color:#ff0000;
  font-size:xx-large;
  }
```

注释："first-letter" 伪元素只能用于块级元素。

注释：下面的属性可应用于 "first-letter" 伪元素：

- font
- color
- background
- margin
- padding
- border
- text-decoration
- vertical-align (仅当 float 为 none 时)
- text-transform
- line-height
- float
- clear

伪元素和 CSS 类

伪元素可以与 CSS 类配合使用：

```
p.article:first-letter
  {
  color: #FF0000;
  }

<p class="article">This is a paragraph in an article。</p>
```

上面的例子会使所有 class 为 article 的段落的首字母变为红色。

多重伪元素

可以结合多个伪元素来使用。

在下面的例子中，段落的第一个字母将显示为红色，其字体大小为 xx-large。第一行中的其余文本将为蓝色，并以小型大写字母显示。段落中的其余文本将以默认字体大小和颜色来显示：

```
p:first-letter
  {
  color:#ff0000;
  font-size:xx-large;
  }

p:first-line
  {
  color:#0000ff;
  font-variant:small-caps;
  }
```

CSS2 - :before 伪元素

":before" 伪元素可以在元素的内容前面插入新内容。

下面的例子在每个 <h1> 元素前面插入一幅图片：

```
h1:before
  {
  content:url(logo.gif);
  }
```

CSS2 - :after 伪元素

":after" 伪元素可以在元素的内容之后插入新内容。

下面的例子在每个 <h1> 元素后面插入一幅图片：

```
h1:after
  {
  content:url(logo.gif);
  }
```

伪元素

W3C："W3C" 列指示出该属性在哪个 CSS 版本中定义（CSS1 还是 CSS2）。

属性	描述	CSS
:first-letter	向文本的第一个字母添加特殊样式。	1
:first-line	向文本的首行添加特殊样式。	1
:before	在元素之前添加内容。	2
:after	在元素之后添加内容。	2

CSS 高级

CSS 水平对齐

在 CSS 中，可以使用多种属性来水平对齐元素。

对齐块元素

块元素指的是占据全部可用宽度的元素，并且在其前后都会换行。

块元素的例子：

```
<h1>
<p>
<div>
```

对于文本对齐，请参见 CSS 文本一章。

在本教程中，我们将向您展示出于布局目的如何水平对齐块级元素。

使用 margin 属性来水平对齐

可通过将左和右外边距设置为 "auto"，来对齐块元素。

注释：除非已经声明了 !DOCTYPE，否则使用 margin:auto 在 IE8 以及更早的版本中是无效的。

把左和右外边距设置为 auto，规定的是均等地分配可用的外边距。结果就是居中的元素：

实例

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

提示：如果宽度是 100%，则对齐没有效果。

注释：在 IE5 中，对于块元素存在一个外边距处理方面的 BUG。如需使上面的例子在 IE5 中有效，请添加一些额外的代码。。

使用 position 属性进行左和右对齐

对齐元素的方法之一是使用绝对定位：

实例

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

注释：绝对定位元素会被从正常流中删除，并且能够交叠元素。

跨浏览器兼容性问题

当像这样对齐元素时，对 <body> 元素的外边距和内边距进行预定义是一个好主意。这样可以避免在不同的浏览器中出现可见的差异。

当使用 position 属性时，IE8 以及更早的版本存在一个问题。如果容器元素（在我们的案例中是 <div class="container">）设置了指定的宽度，并且省略了 !DOCTYPE 声明，那么 IE8 以及更早的版本会在右侧增加 17px 的外边距。这似乎是为滚动条预留的空间。当使用 position 属性时，请始终设置 !DOCTYPE 声明：

实例

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

使用 float 属性来进行左和右对齐

对齐元素的另一种方法是使用 float 属性：

实例

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

跨浏览器兼容性问题

当像这样对齐元素时，对 <body> 元素的外边距和内边距进行预定义是一个好主意。这样可以避免在不同的浏览器中出现可见的差异。

当使用 float 属性时，IE8 以及更早的版本存在一个问题。如果省略 !DOCTYPE 声明，那么 IE8 以及更早的版本会在右侧增加 17px 的外边距。这似乎是为滚动条预留的空间。当使用 float 属性时，请始终设置 !DOCTYPE 声明：

实例

```
body
{
margin:0;
padding:0;
}

.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

CSS 尺寸 (Dimension)

CSS 尺寸 (Dimension) 属性允许你控制元素的高度和宽度。同样，它允许你增加行间距。

CSS 尺寸实例：

使用像素值设置图像的高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img.normal
{
height: auto
}

img.big
{
height: 160px
}

img.small
{
height: 30px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img class="normal" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="big" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="small" src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比设置图像的高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img.normal
{
height: auto
}

img.big
{
height: 50%
}

img.small
{
height: 10%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img class="normal" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="big" src="/i/eg_smile.gif" /&gt;
&lt;br /&gt;
&lt;img class="small" src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
width: 300px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
width: 50%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

设置元素的最大高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-height: 10px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最大宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-width: 300px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的最大宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
max-width: 50%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最小高度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-height: 100px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值来设置元素的最小宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-width: 1000px
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比来设置元素的最小宽度

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p
{
min-width: 200%
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p&gt;这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。
这是一些文本。这是一些文本。这是一些文本。&lt;/p&gt;

&lt;img src="/i/eg_smile.gif" /&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用百分比设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small {line-height: 90%}
p.big {line-height: 200%}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 110% 到 120%。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用像素值设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small
  {
  line-height: 10px
  }
p.big
  {
  line-height: 30px
  }
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
在大多数浏览器中默认行高大约是 20px。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

使用数值来设置行间距

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
p.small
{
line-height: 0.5
}
p.big
{
line-height: 2
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;p&gt;
这是拥有标准行高的段落。
默认行高大约是 1。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
这是拥有标准行高的段落。
&lt;/p&gt;

&lt;p class="small"&gt;
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
这个段落拥有更小的行高。
&lt;/p&gt;

&lt;p class="big"&gt;
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
这个段落拥有更大的行高。
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

CSS 尺寸属性

CSS 尺寸属性允许你控制元素的高度和宽度。同样，还允许你增加行间距。

属性	描述
height	设置元素的高度。
line-height	设置行高。
max-height	设置元素的最大高度。
max-width	设置元素的最大宽度。
min-height	设置元素的最小高度。
min-width	设置元素的最小宽度。
width	设置元素的宽度。

CSS 分类 (Classification)

CSS 分类属性允许你规定如何以及在何处显示元素。

CSS分类(Classification)实例：

如何把元素显示为内联元素

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p {display: inline}
div {display: none}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;本例中的样式表把段落元素设置为内联元素。&lt;/p&gt;

&lt;p&gt;而 div 元素不会显示出来！&lt;/p&gt;

&lt;div&gt;div 元素的内容不会显示出来！&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;
```

如何把元素显示为块级元素

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
span
{
display: block
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;span&gt;本例中的样式表把 span 元素设置为块级元素。&lt;/span&gt;
&lt;span&gt;两个 span 元素之间产生了一个换行行为。&lt;/span&gt;

&lt;/body&gt;
&lt;/html&gt;
```

float 属性的简单应用

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
float:right
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;在下面的段落中，我们添加了一个样式为 &lt;b&gt;float:right&lt;/b&
&lt;p&gt;
&lt;img src="/i/eg_cute.gif" /&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

将带有边框和边界的图像浮动于段落的右侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
img
{
float:right;
border:1px dotted black;
margin:0px 0px 15px 20px;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;在下面的段落中，图像会浮动到右侧，并且添加了点状的边框。我们还为图像
&lt;p&gt;
&lt;img src="/i/eg_cute.gif" /&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

带标题的图像浮动于右侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div
{
float:right;
width:120px;
margin:0 0 15px 20px;
padding:15px;
border:1px solid black;
text-align:center;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;div&gt;
&lt;img src="/i/eg_cute.gif" /&gt;&lt;br /&gt;
CSS is fun!
&lt;/div&gt;
&lt;p&gt;
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;

&lt;p&gt;
在上面的段落中，div 元素的宽度是 120 像素，它其中包含图像。div 元素浮动到右侧
&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

使段落的首字母浮动于左侧

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
span
{
float:left;
width:0.7em;
font-size:400%;
font-family:algerian,courier;
line-height:80%;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;p&gt;
&lt;span&gt;T&lt;/span&gt;his is some text.
This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
&lt;/p&gt;

&lt;p&gt;
在上面的段落中，文本的第一个字母包含在一个 span 元素中。这个 span 元素的宽度是
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建水平菜单

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
ul
{
float:left;
width:100%;
padding:0;
margin:0;
list-style-type:none;
}
a
{
float:left;
width:7em;
text-decoration:none;
color:white;
background-color:purple;
padding:0.2em 0.6em;
border-right:1px solid white;
}
a:hover {background-color:#ff3300}
li {display:inline}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;ul&gt;
&lt;li&gt;&lt;a href="#"&gt;Link one&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link two&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link three&lt;/a&gt;&lt;/li&gt;
&lt;li&gt;&lt;a href="#"&gt;Link four&lt;/a&gt;&lt;/li&gt;
&lt;/ul&gt;

&lt;p&gt;
在上面的例子中，我们把 ul 元素和 a 元素浮向左浮动。li 元素显示为行内元素（元素
&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

创建无表格的首页

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
div.container
{
width:100%;
margin:0px;
```

```
border:1px solid gray;
line-height:150%;
}
div.header,div.footer
{
padding:0.5em;
color:white;
background-color:gray;
clear:left;
}
h1.header
{
padding:0;
margin:0;
}
div.left
{
float:left;
width:160px;
margin:0;
padding:1em;
}
div.content
{
margin-left:190px;
border-left:1px solid gray;
padding:1em;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;div class="container"&gt;

&lt;div class="header"&gt;&lt;h1 class="header"&gt;W3School.com.cn&

&lt;div class="left"&gt;&lt;p&gt;"Never increase, beyond what is ne

&lt;div class="content"&gt;
&lt;h2&gt;Free Web Building Tutorials&lt;/h2&gt;
&lt;p&gt;At W3School.com.cn you will find all the Web-building tuto
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.
&lt;p&gt;W3School.com.cn - The Largest Web Developers Site On The N

&lt;div class="footer"&gt;Copyright 2008 by YingKe Investment.&lt;/
&lt;/div&gt;

&lt;/body&gt;
&lt;/html&gt;
```

定位：相对定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h2.pos_left
{
position:relative;
left:-20px
}
h2.pos_right
{
position:relative;
left:20px
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h2&gt;这是位于正常位置的标题&lt;/h2&gt;
&lt;h2 class="pos_left"&gt;这个标题相对于其正常位置向左移动&lt;/h2&gt;
&lt;h2 class="pos_right"&gt;这个标题相对于其正常位置向右移动&lt;/h2&gt;
&lt;p&gt;相对定位会按照元素的原始位置对该元素进行移动。&lt;/p&gt;
&lt;p&gt;样式 "left:-20px" 从元素的原始左侧位置减去 20 像素。&lt;/p&gt;
&lt;p&gt;样式 "left:20px" 向元素的原始左侧位置增加 20 像素。&lt;/p&gt;
&lt;/body&gt;

&lt;/html&gt;
```

定位：绝对定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h2.pos_abs
{
position:absolute;
left:100px;
top:150px
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h2 class="pos_abs"&gt;这是带有绝对定位的标题&lt;/h2&gt;
&lt;p&gt;通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧
&lt;/body&gt;

&lt;/html&gt;
```

定位：固定定位

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
p.one
{
position:fixed;
left:5px;
top:5px;
}
p.two
{
position:fixed;
top:30px;
right:5px;
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;p class="one"&gt;一些文本。&lt;/p&gt;
&lt;p class="two"&gt;更多的文本。&lt;/p&gt;

&lt;/body&gt;
&lt;/html&gt;
```

如何使元素不可见

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
h1.visible {visibility:visible}
h1.invisible {visibility:hidden}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;h1 class="visible"&gt;这是可见的标题&lt;/h1&gt;
&lt;h1 class="invisible"&gt;这是不可见的标题&lt;/h1&gt;
&lt;/body&gt;

&lt;/html&gt;
```

把表格元素设置为 collapse（请在非 IE 的浏览器中查看）

```
&lt;html&gt;
&lt;head&gt;
&lt;style type="text/css"&gt;
tr.coll
 {
 visibility:collapse
 }
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;table border="1"&gt;
&lt;tr&gt;
&lt;td&gt;Adams&lt;/td&gt;
&lt;td&gt;John&lt;/td&gt;
&lt;/tr&gt;
&lt;tr class="coll"&gt;
&lt;td&gt;Bush&lt;/td&gt;
&lt;td&gt;George&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;

&lt;/body&gt;
&lt;/html&gt;
```

改变光标

```
&lt;html&gt;

&lt;body&gt;
&lt;p&gt;请把鼠标移动到单词上，可以看到鼠标指针发生变化：&lt;/p&gt;
&lt;span style="cursor:auto"&gt;
Auto&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:crosshair"&gt;
Crosshair&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:default"&gt;
Default&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:pointer"&gt;
Pointer&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:move"&gt;
Move&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:e-resize"&gt;
e-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:ne-resize"&gt;
ne-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:nw-resize"&gt;
nw-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:n-resize"&gt;
n-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:se-resize"&gt;
se-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:sw-resize"&gt;
sw-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:s-resize"&gt;
s-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:w-resize"&gt;
w-resize&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:text"&gt;
text&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:wait"&gt;
wait&lt;/span&gt;&lt;br /&gt;
&lt;span style="cursor:help"&gt;
help&lt;/span&gt;
&lt;/body&gt;

&lt;/html&gt;
```

清除元素的侧面

```
&lt;html&gt;

&lt;head&gt;
&lt;style type="text/css"&gt;
img
  {
  float:left;
  clear:both;
  }
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;
&lt;img src="/i/eg_smile.gif" /&gt;
&lt;img src="/i/eg_smile.gif" /&gt;
&lt;/body&gt;

&lt;/html&gt;
```

CSS 分类属性 (Classification)

CSS 分类属性允许你控制如何显示元素，设置图像显示于另一元素中的何处，相对于其正常位置来定位元素，使用绝对值来定位元素，以及元素的可见度。

属性	描述
clear	设置一个元素的侧面是否允许其他的浮动元素。
cursor	规定当指向某元素之上时显示的指针类型。
display	设置是否及如何显示元素。
float	定义元素在哪个方向浮动。
position	把元素放置到一个静态的、相对的、绝对的、或固定的位置中。
visibility	设置元素是否可见或不可见。

CSS 导航条

演示：导航栏

- Home
- News
- Articles
- Forum
- Contact
- About

导航栏

拥有易用的导航条对于任何网站都很重要。

通过 CSS，您能够把乏味的 HTML 菜单转换为漂亮的导航栏。

导航栏 = 链接列表

导航栏需要标准的 HTML 作为基础。

在我们的例子中，将用标准的 HTML 列表来构建导航栏。

导航栏基本上是一个链接列表，因此使用 <ul> 和 <li> 元素是非常合适的：

实例

```
<ul>
<li><a href="default.asp">Home</a></li>
<li><a href="news.asp">News</a></li>
<li><a href="contact.asp">Contact</a></li>
<li><a href="about.asp">About</a></li>
</ul>
```

现在，让我们从列表中去掉圆点和外边距：

实例

```
ul
{
list-style-type:none;
margin:0;
padding:0;
}
```

例子解释：

- list-style-type:none - 删除圆点。导航栏不需要列表项标记。
- 把外边距和内边距设置为 0 可以去除浏览器的默认设定。

以上例子中的代码是用在垂直、水平导航栏中的标准代码。

垂直导航栏

如需构建垂直导航栏，我们只需要定义 <a> 元素的样式，除了上面的代码之外：

实例

```
a
{
display:block;
width:60px;
}
```

例子解释：

- display:block - 把链接显示为块元素可使整个链接区域可点击（不仅仅是文本），同时也允许我们规定宽度。
- width:60px - 块元素默认占用全部可用宽度。我们需要规定 60 像素的宽度。

提示：请同时看一看我们完整样式的导航栏实例。

注释：请始终规定垂直导航栏中 <a> 元素的宽度。如果省略宽度，IE6 会产生意想不到的结果。

水平导航栏

有两种创建水平导航栏的方法。使用行内或浮动列表项。

两种方法都不错，但是如果您希望链接拥有相同的尺寸，就必须使用浮动方法。

行内列表项

除了上面的“标准”代码，构建水平导航栏的方法之一是将 <li> 元素规定为行内元素：

实例

```
li
{
display:inline;
}
```

例子解释：

display:inline; - 默认地， 元素是块元素。在这里，我们去除了每个列表项前后的换行，以便让它们在一行中显示。

提示：请看一下我们完整样式的水平导航栏实例。

对列表项进行浮动

在上面的例子中，链接的宽度是不同的。

为了让所有链接拥有相等的宽度，浮动 <li> 元素并规定 <a> 元素的宽度：

实例

```
li
{
float:left;
}
a
{
display:block;
width:60px;
}
```

例子解释：

- float:left - 使用 float 来把块元素滑向彼此。
- display:block - 把链接显示为块元素可使整个链接区域可点击（不仅仅是文本），同时也允许我们规定宽度。
- width:60px - 由于块元素默认占用全部可用宽度，链接无法滑动至彼此相邻。我们需要规定 60 像素的宽度。

提示：请看一下我们完整样式的水平导航栏实例。

CSS 图片库

CSS 可用来创建图片库。

演示：CSS 图片库


在此处添加对图像的描述


在此处添加对图像的描述


在此处添加对图像的描述




在此处添加对图像的描述

图片库

下面的图片库是由 CSS 创建的：

实例

```
<!doctype html>
<html>
<head>
<style>
div.img
  {
  margin:3px;
  border:1px solid #bebebe;
  height:auto;
  width:auto;
  float:left;
  text-align:center;
  }
div.img img
```

```
  {
  display:inline;
  margin:3px;
  border:1px solid #bebebe;
  }
div.img a:hover img
  {
  border:1px solid #333333;
  }
div.desc
  {
  text-align:center;
  font-weight:normal;
  width:150px;
  font-size:12px;
  margin:10px 5px 10px 5px;
  }
</style>
</head>
<body>

<div class="img">
  <a target="_blank" href="/i/tulip_ballade.jpg">
  <img src="/i/tulip_ballade_s.jpg" alt="Ballade" width="160" heigh
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_flaming_club.jpg">
  <img src="/i/tulip_flaming_club_s.jpg" alt="Ballade" width="160"
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_fringed_family.jpg">
  <img src="/i/tulip_fringed_family_s.jpg" alt="Ballade" width="160
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

<div class="img">
  <a target="_blank" href="/i/tulip_peach_blossom.jpg">
  <img src="/i/tulip_peach_blossom_s.jpg" alt="Ballade" width="160"
  </a>
  <div class="desc">在此处添加对图像的描述</div>
</div>

</body>
</html>
```

CSS 图像透明度

通过 CSS 创建透明图像是很容易的。

注释：CSS opacity 属性是 W3C CSS 推荐标准的一部分。

亲自试一试 - 实例

创建透明图像 - Hover 效果

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style&gt;
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* For IE8 and earlier */
}
&lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;

&lt;h1&gt;图像透明度&lt;/h1&gt;

&lt;img src="/i/tulip_peach_blossom_w_s.jpg" alt="Peach Blossom" /&

&lt;p&gt;&lt;b&gt;注释：&lt;/b&gt;在 IE 中，必须添加 &lt;!DOCTYPE&gt;,
&lt;/body&gt;
&lt;/html&gt;
```

创建文本在背景图像上的透明框

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
&lt;style&gt;
div.background
{
  width: 400px;
```

```
  height: 266px;
  margin:15px;
  background: url('/i/tulip_peach_blossom_w.jpg') no-repeat;
  border: 1px solid black;
}

div.transbox
{
  width: 338px;
  height: 204px;
  margin:30px;
  padding:0;
  background-color: #ffffff;
  border: 1px solid black;
  /* for IE */
  filter:alpha(opacity=60);
  /* CSS3 standard */
  opacity:0.6;
}

div.transbox p
{
  margin: 30px 40px;
  color: #000000;
  font:bold 12px Verdana, Geneva, sans-serif;
  line-height:1.5;
}
&lt;/style&gt;
&lt;/head&gt;

&lt;body&gt;

&lt;div class="background"&gt;
&lt;div class="transbox"&gt;
&lt;p&gt;
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
&lt;/p&gt;
&lt;/div&gt;
&lt;/div&gt;

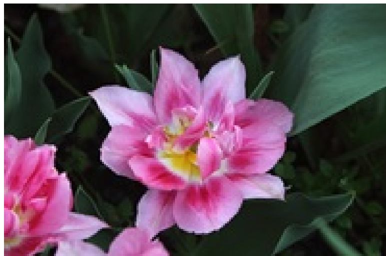
&lt;/body&gt;
&lt;/html&gt;
```

实例 1 - 创建透明图像

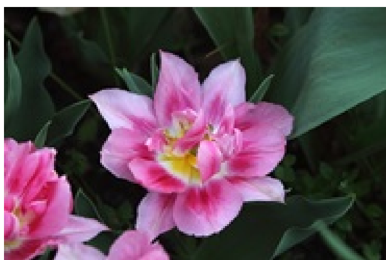
定义透明效果的 CSS3 属性是 *opacity*。

首先，我们将展示如何通过 CSS 来创建透明图像。

常规图像：



带有透明度的相同图像：



请看下面的 CSS：

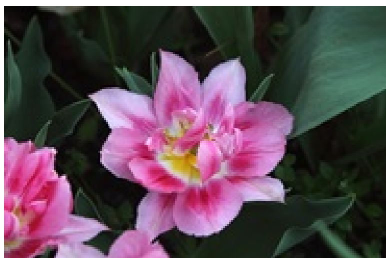
```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* 针对 IE8 以及更早的版本 */
}
```

IE9, Firefox, Chrome, Opera 和 Safari 使用属性 *opacity* 来设定透明度。opacity 属性能够设置的值从 0.0 到 1.0。值越小，越透明。

IE8 以及更早的版本使用滤镜 *filter:alpha(opacity=x)*。x 能够取的值从 0 到 100。值越小，越透明。

实例 2 - 图像透明度 - Hover 效果

请把鼠标指针移动到图像上：



CSS 是这样的：

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* 针对 IE8 以及更早的版本 */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* 针对 IE8 以及更早的版本 */
}
```

第一个 CSS 代码块类似实例 1 中的代码。此外，我们已经设置了当鼠标指针位于图像上时的样式。在这个例子中，当指针移动到图像上时，我们希望图像是不透明的。

对应的 CSS 是：*opacity=1*。

IE8 以及更早的浏览器：*filter:alpha(opacity=100)*。

当鼠标指针移出图像后，图像会再次透明。

实例 3 - 透明框中的文本

This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box.

源代码是这样的：

```
<!DOCTYPE html>
<html>
<head>
<style>
div.background
{
  width: 400px;
  height: 266px;
  background: url('/i/tulip_peach_blossom_w.jpg') no-repeat;
  border: 1px solid black;
}

div.transbox
{
  width: 338px;
  height: 204px;
  margin:30px;
  background-color: #ffffff;
  border: 1px solid black;
  /* for IE */
  filter:alpha(opacity=60);
  /* CSS3 standard */
  opacity:0.6;
}

div.transbox p
{
  margin: 30px 40px;
}
</style>
</head>

<body>

<div class="background">
<div class="transbox">
<p>
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>

</body>
</html>
```

首先，我们创建一个 div 元素 (class="background")，它有固定的高度和宽度、背景图像，以及边框。然后我们在第一个 div 内创建稍小的 div (class="transbox")。"transbox" div 有固定的宽度、背景色和边框 - 并且它是透明的。在透明 div 内部，我们在 p 元素中加入了一些文本。

相关页面

CSS 参考手册：CSS3 opacity 属性

CSS2 媒介类型

媒介类型**(Media Types)**允许你定义以何种媒介来提交文档。文档可以被显示在显示器、纸媒介或者听觉浏览器等等。

媒介类型

某些 CSS 属性仅仅被设计为针对某些媒介。比方说 "voice-family" 属性被设计为针对听觉用户终端。其他的属性可被用于不同的媒介。例如，"font-size" 属性可被用于显示器以及印刷媒介，但是也许会带有不同的值。显示器上面的显示的文档通常会需要比纸媒介文档更大的字号，同时，在显示器上，sans-serif 字体更易阅读，而在纸媒介上，serif 字体更易阅读。

@media规则

@media 规则使你有能力在相同的样式表中，使用不同的样式规则来针对不同的媒介。

下面这个例子中的样式告知浏览器在显示器上显示 14 像素的 Verdana 字体。但是假如页面需要被打印，将使用 10 个像素的 Times 字体。注意：font-weight 被设置为粗体，不论显示器还是纸媒介：

```
<html>
<head>

<style>
@media screen
{
p.test {font-family:verdana,sans-serif; font-size:14px}
}

@media print
{
p.test {font-family:times,serif; font-size:10px}
}

@media screen,print
{
p.test {font-weight:bold}
}
</style>

</head>

<body>....</body>

</html>
```

不同的媒介类型

注释：媒介类型名称对大小写不敏感。

媒介类型	描述
all	用于所有的媒介设备。
aural	用于语音和音频合成器。
braille	用于盲人用点字法触觉回馈设备。
embossed	用于分页的盲人用点字法打印机。
handheld	用于小的手持的设备。
print	用于打印机。
projection	用于方案展示，比如幻灯片。
screen	用于电脑显示器。
tty	用于使用固定密度字母栅格的媒介，比如电传打字机和终端。
tv	用于电视机类型的设备。

CSS 注意事项

本节列出了在使用 **CSS** 时尽量避免使用的技术。

Internet Explorer Behaviors

_它是什么？_Internet Explorer 5 引入了行为 (behaviors)。behaviors 是一种通过使用 CSS 向 HTML 元素添加行为的方法。

为什么要避免它？只有 Internet Explorer 支持 behavior 属性。

用什么代替？请使用 JavaScript 和 HTML DOM 取而代之。

例子 1 - Mouseover Highlight

下面的 HTML 文件中有一个 <style> 元素，它为 <h1> 元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
h1 { behavior: url(behave.htc) }
</style>
</head>

<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

下面是 XML 文档 "behave.htc"：

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}

function low_lite()
{
element.style.color='blue';
}
</script>
```

behavior 文件包含了针对元素的 JavaScript 和 事件句柄。

如果您使用 Internet Explorer，可以亲自试一下（把鼠标放在例子中的文本上）。

例子 2 - Typewriter Simulation

下面的 HTML 文件中有一个 <style> 元素，它为 id 为 "typing" 的元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
#typing
{
behavior:url(behave_typing.htc);
font-family:'courier new';
}
</style>
</head>

<body>
<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
</span>
</body>
</html>
```

下面是 XML 文档 "behave.htc"：

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
  {
  clearInterval(t);
  }
}
</script>
```

如果您使用 Internet Explorer，可以亲自试一下。

CSS 高级

CSS3 简介

CSS3 完全向后兼容，因此您不必改变现有的设计。浏览器通常支持 CSS2。

CSS3 模块

CSS3 被划分为模块。

其中最重要的 CSS3 模块包括：

- 选择器
- 框模型
- 背景和边框
- 文本效果
- 2D/3D 转换
- 动画
- 多列布局
- 用户界面

CSS3 标准

W3C 仍然在对 CSS3 规范进行开发。

不过，现代浏览器已经实现了相当多的 CSS3 属性。

CSS3 边框

CSS3 边框

通过 CSS3，您能够创建圆角边框，向矩形添加阴影，使用图片来绘制边框 - 并且不需使用设计软件，比如 PhotoShop。

在本章中，您将学到以下边框属性：

- border-radius
- box-shadow
- border-image

浏览器支持

属性	浏览器支持
border-radius	
box-shadow	
border-image	

Internet Explorer 9+ 支持 border-radius 和 box-shadow 属性。

Firefox、Chrome 以及 Safari 支持所有新的边框属性。

注释：对于 border-image，Safari 5 以及更老的版本需要前缀 -webkit-。

Opera 支持 border-radius 和 box-shadow 属性，但是对于 border-image 需要前缀 -o-。

CSS3 圆角边框

在 CSS2 中添加圆角矩形需要技巧。我们必须为每个圆角使用不同的图片。

在 CSS3 中，创建圆角是非常容易的。

在 CSS3 中，border-radius 属性用于创建圆角：

这个矩形有圆角哦！

实例

向 div 元素添加圆角：

```
div
{
border:2px solid;
border-radius:25px;
-moz-border-radius:25px; /* Old Firefox */
}
```

CSS3 边框阴影

在 CSS3 中，box-shadow 用于向方框添加阴影：

实例

向 div 元素添加方框阴影：

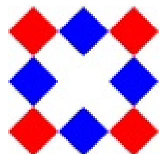
```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

CSS3 边框图片

通过 CSS3 的 border-image 属性，您可以使用图片来创建边框：

border-image 属性允许您规定用于边框的图片！

用于创建上面的边框的原始图片：



实例

使用图片创建围绕 div 元素的边框：

```
div
{
border-image:url(border.png) 30 30 round;
-moz-border-image:url(border.png) 30 30 round; /* 老的 Firefox */
-webkit-border-image:url(border.png) 30 30 round; /* Safari 和 Chro
-o-border-image:url(border.png) 30 30 round; /* Opera */
}
```

新的边框属性

属性	描述	CSS
border-image	设置所有 border-image-* 属性的简写属性。	3
border-radius	设置所有四个 border-*-radius 属性的简写属性。	3
box-shadow	向方框添加一个或多个阴影。	3

CSS3 背景

CSS3 背景

CSS3 包含多个新的背景属性，它们提供了对背景更强大的控制。

在本章，您将学到以下背景属性：

- background-size
- background-origin

您也将学到如何使用多重背景图片。

浏览器支持

属性	浏览器支持
background-size	
background-origin	

Internet Explorer 9+、Firefox、Chrome、Safari 以及 Opera 支持新的背景属性。

CSS3 background-size 属性

background-size 属性规定背景图片的尺寸。

在 CSS3 之前，背景图片的尺寸是由图片的实际尺寸决定的。在 CSS3 中，可以规定背景图片的尺寸，这就允许我们在不同的环境中重复使用背景图片。

您能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸相对于父元素的宽度和高度。

例子 1

调整背景图片的大小：

```
div
{
background:url(bg_flower.gif);
-moz-background-size:63px 100px; /* 老版本的 Firefox */
background-size:63px 100px;
background-repeat:no-repeat;
}
```

例子 2

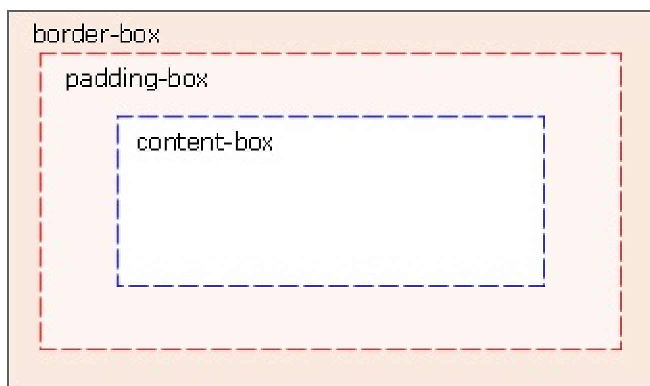
对背景图片进行拉伸，使其完成填充内容区域：

```
div
{
background:url(bg_flower.gif);
-moz-background-size:40% 100%; /* 老版本的 Firefox */
background-size:40% 100%;
background-repeat:no-repeat;
}
```

CSS3 background-origin 属性

background-origin 属性规定背景图片的定位区域。

背景图片可以放置于 content-box、padding-box 或 border-box 区域。



实例

在 content-box 中定位背景图片：

```
div
{
background:url(bg_flower.gif);
background-repeat:no-repeat;
background-size:100% 100%;
-webkit-background-origin:content-box; /* Safari */
background-origin:content-box;
}
```

CSS3 多重背景图片

CSS3 允许您为元素使用多个背景图像。

实例

为 body 元素设置两幅背景图片：

```
body
{
background-image:url(bg_flower.gif),url(bg_flower_2.gif);
}
```

新的背景属性

属性	描述	CSS
background-clip	规定背景的绘制区域。	3
background-origin	规定背景图片的定位区域。	3
background-size	规定背景图片的尺寸。	3

CSS3 文本效果

CSS3 文本效果

CSS3 包含多个新的文本特性。

在本章中，您将学到如下文本属性：

- text-shadow
- word-wrap

浏览器支持

属性	浏览器支持
text-shadow	
word-wrap	

Internet Explorer 10、Firefox、 Chrome、Safari 以及 Opera 支持 text-shadow 属性。

所有主流浏览器都支持 word-wrap 属性。

注释：Internet Explorer 9 以及更早的版本，不支持 text-shadow 属性。

CSS3 文本阴影

在 CSS3 中，text-shadow 可向文本应用阴影。

文本阴影效果！

您能够规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色：

实例

向标题添加阴影：

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

CSS3 自动换行

单词太长的话就可能无法超出某个区域：

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

在 CSS3 中，word-wrap 属性允许您允许文本强制文本进行换行 - 即使这意味着会对单词进行拆分：

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

下面是 CSS 代码：

实例

允许对长单词进行拆分，并换行到下一行：

```
p {word-wrap:break-word;}
```

新的文本属性

属性	描述	CSS
hanging-punctuation	规定标点字符是否位于线框之外。	3
punctuation-trim	规定是否对标点字符进行修剪。	3
text-align-last	设置如何对齐最后一行或紧挨着强制换行符之前的行。	3
text-emphasis	向元素的文本应用重点标记以及重点标记的前景色。	3
text-justify	规定当 text-align 设置为 "justify" 时所使用的对齐方法。	3
text-outline	规定文本的轮廓。	3
text-overflow	规定当文本溢出包含元素时发生的事情。	3
text-shadow	向文本添加阴影。	3
text-wrap	规定文本的换行规则。	3
word-break	规定非中日韩文本的换行规则。	3
word-wrap	允许对长的不可分割的单词进行分割并换行到下一行。	3

CSS3 字体

通过 CSS3，Web 设计师再也不必被迫使用“web-safe”字体了。

CSS3 @font-face 规则

在 CSS3 之前，web 设计师必须使用已在用户计算机上安装好的字体。

通过 CSS3，web 设计师可以使用他们喜欢的任意字体。

当您您找到或购买到希望使用的字体时，可将该字体文件存放到 web 服务器上，它会在需要时被自动下载到用户的计算机上。

您“自己的”的字体是在 CSS3 @font-face 规则中定义的。

浏览器支持

属性	浏览器支持
@font-face	

Firefox、Chrome、Safari 以及 Opera 支持 .ttf (True Type Fonts) 和 .otf (OpenType Fonts) 类型的字体。

Internet Explorer 9+ 支持新的 @font-face 规则，但是仅支持 .eot 类型的字体 (Embedded OpenType)。

注释：Internet Explorer 8 以及更早的版本不支持新的 @font-face 规则。

使用您需要的字体

在新的 @font-face 规则中，您必须首先定义字体的名称（比如 myFirstFont），然后指向该字体文件。

如需为 HTML 元素使用字体，请通过 font-family 属性来引用字体的名称 (myFirstFont)：

实例

```
<style>
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Light.ttf'),
     url('Sansation_Light.eot'); /* IE9+ */
}

div
{
font-family:myFirstFont;
}
</style>
```

使用粗体字体

您必须为粗体文本添加另一个包含描述符的 @font-face：

实例

```
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Bold.ttf'),
     url('Sansation_Bold.eot'); /* IE9+ */
font-weight:bold;
}
```

文件 "Sansation_Bold.ttf" 是另一个字体文件，它包含了 Sansation 字体的粗体字符。

只要 font-family 为 "myFirstFont" 的文本需要显示为粗体，浏览器就会使用该字体。

通过这种方式，我们可以为相同的字体设置许多 @font-face 规则。

CSS3 字体描述符

下面的表格列出了能够在 @font-face 规则中定义的所有字体描述符：

描述符	值	描述
font-family	*name*	必需。规定字体的名称。
src	*URL*	必需。定义字体文件的 URL。
font-stretch	normal condensed ultra-condensed extra-condensed semi-condensed expanded semi-expanded extra-expanded ultra-expanded	可选。定义如何拉伸字体。默认是 "normal"。
font-style	ormal italic oblique	可选。定义字体的样式。默认是 "normal"。
font-weight	normal bold 100 200 300 400 500 600 700 800 900	可选。定义字体的粗细。默认是 "normal"。
unicode-range	*unicode-range*	可选。定义字体支持的 UNICODE 字符范围。默认是 "U+0-10FFFF"。

CSS3 2D 转换

CSS3 转换

通过 CSS3 转换，我们能够对元素进行移动、缩放、转动、拉长或拉伸。

它如何工作？

转换是使元素改变形状、尺寸和位置的一种效果。

您可以使用 2D 或 3D 转换来转换您的元素。

浏览器支持

属性	浏览器支持
transform	

Internet Explorer 10、Firefox 以及 Opera 支持 transform 属性。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 需要前缀 -ms-。

2D 转换

在本章中，您将学到如下 2D 转换方法：

- translate()
- rotate()
- scale()
- skew()
- matrix()

您将在下一章学习 3D 转换。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);         /* IE 9 */
-webkit-transform: rotate(30deg);     /* Safari and Chrome */
-o-transform: rotate(30deg);         /* Opera */
-moz-transform: rotate(30deg);        /* Firefox */
}
```

translate() 方法

通过 translate() 方法，元素从其当前位置移动，根据给定的 left（x 坐标） 和 top（y 坐标） 位置参数：

实例

```
div
{
transform: translate(50px,100px);
-ms-transform: translate(50px,100px);         /* IE 9 */
-webkit-transform: translate(50px,100px);     /* Safari and Chrome 
-o-transform: translate(50px,100px);         /* Opera */
-moz-transform: translate(50px,100px);        /* Firefox */
}
```

值 translate(50px,100px) 把元素从左侧移动 50 像素，从顶端移动 100 像素。

rotate() 方法

通过 rotate() 方法，元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。

实例

```
div
{
transform: rotate(30deg);
-ms-transform: rotate(30deg);         /* IE 9 */
-webkit-transform: rotate(30deg);     /* Safari and Chrome */
-o-transform: rotate(30deg);         /* Opera */
-moz-transform: rotate(30deg);        /* Firefox */
}
```

值 rotate(30deg) 把元素顺时针旋转 30 度。

scale() 方法

通过 scale() 方法，元素的尺寸会增加或减少，根据给定的宽度（X 轴）和高度（Y 轴）参数：

实例

```
div
{
transform: scale(2,4);
-ms-transform: scale(2,4);    /* IE 9 */
-webkit-transform: scale(2,4);    /* Safari 和 Chrome */
-o-transform: scale(2,4);    /* Opera */
-moz-transform: scale(2,4);    /* Firefox */
}
```

值 scale(2,4) 把宽度转换为原始尺寸的 2 倍，把高度转换为原始高度的 4 倍。

skew() 方法

通过 skew() 方法，元素翻转给定的角度，根据给定的水平线（X 轴）和垂直线（Y 轴）参数：

实例

```
div
{
transform: skew(30deg,20deg);
-ms-transform: skew(30deg,20deg);    /* IE 9 */
-webkit-transform: skew(30deg,20deg);    /* Safari and Chrome */
-o-transform: skew(30deg,20deg);    /* Opera */
-moz-transform: skew(30deg,20deg);    /* Firefox */
}
```

值 skew(30deg,20deg) 围绕 X 轴把元素翻转 30 度，围绕 Y 轴翻转 20 度。

matrix() 方法

matrix() 方法把所有 2D 转换方法组合在一起。

matrix() 方法需要六个参数，包含数学函数，允许您：旋转、缩放、移动以及倾斜元素。

实例

如何使用 matrix 方法将 div 元素旋转 30 度：

```
div
{
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0);        /* IE 9 */
-moz-transform:matrix(0.866,0.5,-0.5,0.866,0,0);    /* Firefox */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0);    /* Safari ar
-o-transform:matrix(0.866,0.5,-0.5,0.866,0,0);        /* Opera */
}
```

新的转换属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3

2D Transform 方法

函数	描述
matrix(*n*,*n*,*n*,*n*,*n*,*n*)	定义 2D 转换，使用六个值的矩阵。
translate(*x*,*y*)	定义 2D 转换，沿着 X 和 Y 轴移动元素。
translateX(*n*)	定义 2D 转换，沿着 X 轴移动元素。
translateY(*n*)	定义 2D 转换，沿着 Y 轴移动元素。
scale(*x*,*y*)	定义 2D 缩放转换，改变元素的宽度和高度。
scaleX(*n*)	定义 2D 缩放转换，改变元素的宽度。
scaleY(*n*)	定义 2D 缩放转换，改变元素的高度。
rotate(*angle*)	定义 2D 旋转，在参数中规定角度。
skew(*x-angle*,*y-angle*)	定义 2D 倾斜转换，沿着 X 和 Y 轴。
skewX(*angle*)	定义 2D 倾斜转换，沿着 X 轴。
skewY(*angle*)	定义 2D 倾斜转换，沿着 Y 轴。

CSS3 3D 转换

3D 转换

CSS3 允许您使用 3D 转换来对元素进行格式化。

在本章中，您将学到其中的一些 3D 转换方法：

- rotateX()
- rotateY()

点击下面的元素，来查看 2D 转换与 3D 转换之间的不同之处：

2D 旋转

3D 旋转

它如何工作？

转换是使元素改变形状、尺寸和位置的一种效果。

您可以使用 2D 或 3D 转换来转换您的元素。

浏览器支持

属性	浏览器支持
transform	

Internet Explorer 10 和 Firefox 支持 3D 转换。

Chrome 和 Safari 需要前缀 -webkit-。

Opera 仍然不支持 3D 转换（它只支持 2D 转换）。

rotateX() 方法

通过 rotateX() 方法，元素围绕其 X 轴以给定的度数进行旋转。

实例

```
div
{
transform: rotateX(120deg);
-webkit-transform: rotateX(120deg);	/* Safari 和 Chrome */
-moz-transform: rotateX(120deg);	/* Firefox */
}
```

rotateY() 旋转

通过 rotateY() 方法，元素围绕其 Y 轴以给定的度数进行旋转。

实例

```
div
{
transform: rotateY(130deg);
-webkit-transform: rotateY(130deg);	/* Safari 和 Chrome */
-moz-transform: rotateY(130deg);	/* Firefox */
}
```

转换属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transform	向元素应用 2D 或 3D 转换。	3
transform-origin	允许你改变被转换元素的位置。	3
transform-style	规定被嵌套元素如何在 3D 空间中显示。	3
perspective	规定 3D 元素的透视效果。	3
perspective-origin	规定 3D 元素的底部位置。	3
backface-visibility	定义元素在不面对屏幕时是否可见。	3

2D Transform 方法

函数	描述
matrix3d(*n*,*n*,*n*,*n*,*n*,*n*, *n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*,*n*)	定义 3D 转换，使用 16 个值的 4x4 矩阵。
translate3d(*x*,*y*,*z*)	定义 3D 转化。
translateX(*x*)	定义 3D 转化，仅使用用于 X 轴的值。
translateY(*y*)	定义 3D 转化，仅使用用于 Y 轴的值。
translateZ(*z*)	定义 3D 转化，仅使用用于 Z 轴的值。
scale3d(*x*,*y*,*z*)	定义 3D 缩放转换。
scaleX(*x*)	定义 3D 缩放转换，通过给定一个 X 轴的值。
scaleY(*y*)	定义 3D 缩放转换，通过给定一个 Y 轴的值。
scaleZ(*z*)	定义 3D 缩放转换，通过给定一个 Z 轴的值。
rotate3d(*x*,*y*,*z*,*angle*)	定义 3D 旋转。
rotateX(*angle*)	定义沿 X 轴的 3D 旋转。
rotateY(*angle*)	定义沿 Y 轴的 3D 旋转。
rotateZ(*angle*)	定义沿 Z 轴的 3D 旋转。
perspective(*n*)	定义 3D 转换元素的透视视图。

CSS3 过渡

CSS3 过渡

通过 CSS3，我们可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。

请把鼠标移动到下面的元素上：

CSS3 过渡

浏览器支持

属性	浏览器支持
transition	

Internet Explorer 10、Firefox、Chrome 以及 Opera 支持 transition 属性。

Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 以及更早的版本，不支持 transition 属性。

注释：Chrome 25 以及更早的版本，需要前缀 -webkit-。

它如何工作？

CSS3 过渡是元素从一种样式逐渐改变为另一种的效果。

要实现这一点，必须规定两项内容：

- 规定您希望把效果添加到哪个 CSS 属性上
- 规定效果的时长

实例

应用于宽度属性的过渡效果，时长为 2 秒：

```
div
{
transition: width 2s;
-moz-transition: width 2s;    /* Firefox 4 */
-webkit-transition: width 2s;    /* Safari 和 Chrome */
-o-transition: width 2s;    /* Opera */
}
```

注释：如果时长未规定，则不会有过渡效果，因为默认值是 0。

效果开始于指定的 CSS 属性改变值时。CSS 属性改变的典型时间是鼠标指针位于元素上时：

实例

规定当鼠标指针悬浮于 <div> 元素上时：

```
div:hover
{
width:300px;
}
```

注释：当指针移出元素时，它会逐渐变回原来的样式。

多项改变

如需向多个样式添加过渡效果，请添加多个属性，由逗号隔开：

实例

向宽度、高度和转换添加过渡效果：

```
div
{
transition: width 2s, height 2s, transform 2s;
-moz-transition: width 2s, height 2s, -moz-transform 2s;
-webkit-transition: width 2s, height 2s, -webkit-transform 2s;
-o-transition: width 2s, height 2s,-o-transform 2s;
}
```

过渡属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。默认是 0。	3
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。	3
transition-delay	规定过渡效果何时开始。默认是 0。	3

下面的两个例子设置所有过渡属性：

实例

在一个例子中使用所有过渡属性：

```
div
{
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
/* Firefox 4 */
-moz-transition-property:width;
-moz-transition-duration:1s;
-moz-transition-timing-function:linear;
-moz-transition-delay:2s;
/* Safari 和 Chrome */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
/* Opera */
-o-transition-property:width;
-o-transition-duration:1s;
-o-transition-timing-function:linear;
-o-transition-delay:2s;
}
```

实例

与上面的例子相同的过渡效果，但是使用了简写的 transition 属性：

```
div
{
transition: width 1s linear 2s;
/* Firefox 4 */
-moz-transition:width 1s linear 2s;
/* Safari and Chrome */
-webkit-transition:width 1s linear 2s;
/* Opera */
-o-transition:width 1s linear 2s;
}
```

CSS3 动画

CSS3 动画

通过 CSS3，我们能够创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。

CSS3 动画

CSS3 @keyframes 规则

如需在 CSS3 中创建动画，您需要学习 @keyframes 规则。

@keyframes 规则用于创建动画。在 @keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

浏览器支持

属性	浏览器支持
@keyframes	
animation	

Internet Explorer 10、Firefox 以及 Opera 支持 @keyframes 规则和 animation 属性。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9，以及更早的版本，不支持 @keyframe 规则或 animation 属性。

实例

```
@keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}

@-moz-keyframes myfirst /* Firefox */
{
from {background: red;}
to {background: yellow;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
from {background: red;}
to {background: yellow;}
}

@-o-keyframes myfirst /* Opera */
{
from {background: red;}
to {background: yellow;}
}
```

CSS3 动画

当您在 @keyframes 中创建动画时，请把它捆绑到某个选择器，否则不会产生动画效果。

通过规定至少以下两项 CSS3 动画属性，即可将动画绑定到选择器：

- 规定动画的名称
- 规定动画的时长

实例

把 "myfirst" 动画捆绑到 div 元素，时长：5 秒：

```
div
{
animation: myfirst 5s;
-moz-animation: myfirst 5s;	/* Firefox */
-webkit-animation: myfirst 5s;	/* Safari 和 Chrome */
-o-animation: myfirst 5s;	/* Opera */
}
```

注释：您必须定义动画的名称和时长。如果忽略时长，则动画不会允许，因为默认值是 0。

什么是 CSS3 中的动画？

动画是使元素从一种样式逐渐变化为另一种样式的效果。

您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

0% 是动画的开始，100% 是动画的完成。

为了得到最佳的浏览器支持，您应该始终定义 0% 和 100% 选择器。

实例

当动画为 25% 及 50% 时改变背景色，然后当动画 100% 完成时再次改变：

```
@keyframes myfirst
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-moz-keyframes myfirst /* Firefox */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

@-o-keyframes myfirst /* Opera */
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}
```

实例

改变背景色和位置：

```
@keyframes myfirst
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-moz-keyframes myfirst /* Firefox */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-webkit-keyframes myfirst /* Safari 和 Chrome */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

@-o-keyframes myfirst /* Opera */
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

CSS3 动画属性

下面的表格列出了 @keyframes 规则和所有动画属性：

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。	3
animation-timing-function	规定动画的速度曲线。默认是 "ease"。	3
animation-delay	规定动画何时开始。默认是 0。	3
animation-iteration-count	规定动画被播放的次数。默认是 1。	3
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。	3
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。	3
animation-fill-mode	规定对象动画时间之外的状态。	3

下面的两个例子设置了所有动画属性：

实例

运行名为 myfirst 的动画，其中设置了所有动画属性：

```
div
{
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
/* Firefox: */
-moz-animation-name: myfirst;
-moz-animation-duration: 5s;
-moz-animation-timing-function: linear;
-moz-animation-delay: 2s;
-moz-animation-iteration-count: infinite;
-moz-animation-direction: alternate;
-moz-animation-play-state: running;
/* Safari 和 Chrome: */
-webkit-animation-name: myfirst;
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
/* Opera: */
-o-animation-name: myfirst;
-o-animation-duration: 5s;
-o-animation-timing-function: linear;
-o-animation-delay: 2s;
-o-animation-iteration-count: infinite;
-o-animation-direction: alternate;
-o-animation-play-state: running;
}
```

实例

与上面的动画相同，但是使用了简写的动画 animation 属性：

```
div
{
animation: myfirst 5s linear 2s infinite alternate;
/* Firefox: */
-moz-animation: myfirst 5s linear 2s infinite alternate;
/* Safari 和 Chrome: */
-webkit-animation: myfirst 5s linear 2s infinite alternate;
/* Opera: */
-o-animation: myfirst 5s linear 2s infinite alternate;
}
```

CSS3 多列

CSS3 多列

通过 CSS3，您能够创建多个列来对文本进行布局 - 就像报纸那样！

在本章中，您将学习如下多列属性：

- column-count
- column-gap
- column-rule

浏览器支持

属性	浏览器支持
column-count	
column-gap	
column-rule	

Internet Explorer 10 和 Opera 支持多列属性。

Firefox 需要前缀 -moz-。

Chrome 和 Safari 需要前缀 -webkit-。

注释：Internet Explorer 9 以及更早的版本不支持多列属性。

CSS3 创建多列

column-count 属性规定元素应该被分隔的列数：

实例

把 div 元素中的文本分隔为三列：

```
div
{
-moz-column-count:3;     /* Firefox */
-webkit-column-count:3; /* Safari 和 Chrome */
column-count:3;
}
```

CSS3 规定列之间的间隔

column-gap 属性规定列之间的间隔：

实例

规定列之间 40 像素的间隔：

```
div
{
-moz-column-gap:40px;        /* Firefox */
-webkit-column-gap:40px;    /* Safari 和 Chrome */
column-gap:40px;
}
```

CSS3 列规则

column-rule 属性设置列之间的宽度、样式和颜色规则。

实例

规定列之间的宽度、样式和颜色规则：

```
div
{
-moz-column-rule:3px outset #ff0000;    /* Firefox */
-webkit-column-rule:3px outset #ff0000;    /* Safari and Chrome */
column-rule:3px outset #ff0000;
}
```

新的多列属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
column-count	规定元素应该被分隔的列数。	3
column-fill	规定如何填充列。	3
column-gap	规定列之间的间隔。	3
column-rule	设置所有 column-rule-* 属性的简写属性。	3
column-rule-color	规定列之间规则的颜色。	3
column-rule-style	规定列之间规则的样式。	3
column-rule-width	规定列之间规则的宽度。	3
column-span	规定元素应该横跨的列数。	3
column-width	规定列的宽度。	3
columns	规定设置 column-width 和 column-count 的简写属性。	3

CSS3 用户界面

CSS3 用户界面

在 CSS3 中，新的用户界面特性包括重设元素尺寸、盒尺寸以及轮廓等。

在本章中，您将学到以下用户界面属性：

- resize
- box-sizing
- outline-offset

浏览器支持

属性	浏览器支持
resize	
box-sizing	
outline-offset	

Firefox、Chrome 以及 Safari 支持 resize 属性。

Internet Explorer、Chrome、Safari 以及 Opera 支持 box-sizing 属性。Firefox 需要前缀 -moz-。

所有主流浏览器都支持 outline-offset 属性，除了 Internet Explorer。

CSS3 Resizing

在 CSS3，resize 属性规定是否可由用户调整元素尺寸。

这个 div 元素可由用户调整尺寸（在 Firefox 4+、Chrome 以及 Safari 中）。

CSS 代码如下：

实例

规定 div 元素可由用户调整大小：

```
div
{
resize:both;
overflow:auto;
}
```

CSS3 Box Sizing

box-sizing 属性允许您以确切的方式定义适应某个区域的具体内容。

实例

规定两个并排的带边框方框：

```
div
{
box-sizing:border-box;
-moz-box-sizing:border-box;    /* Firefox */
-webkit-box-sizing:border-box;    /* Safari */
width:50%;
float:left;
}
```

CSS3 Outline Offset

outline-offset 属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

轮廓与边框有两点不同：

- 轮廓不占用空间
- 轮廓可能是非矩形

这个 div 在边框之外 15 像素处有一个轮廓。

CSS 代码如下：

实例

规定边框边缘之外 15 像素处的轮廓：

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

新的用户界面属性

下面的表格列出了所有的转换属性：

属性	描述	CSS
appearance	允许您将元素设置为标准用户界面元素的外观	3
box-sizing	允许您以确切的方式定义适应某个区域的具体内容。	3
icon	为创作者提供使用图标化等价物来设置元素样式的能力。	3
nav-down	规定在使用 arrow-down 导航键时向何处导航。	3
nav-index	设置元素的 tab 键控制次序。	3
nav-left	规定在使用 arrow-left 导航键时向何处导航。	3
nav-right	规定在使用 arrow-right 导航键时向何处导航。	3
nav-up	规定在使用 arrow-up 导航键时向何处导航。	3
outline-offset	对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。	3
resize	规定是否可由用户对元素的尺寸进行调整。	3

Firebug 教程

Firebug 教程

什么是 Firebug？

Firebug 是一个开源的web开发工具。

在本章教程我们将讨论以下内容：

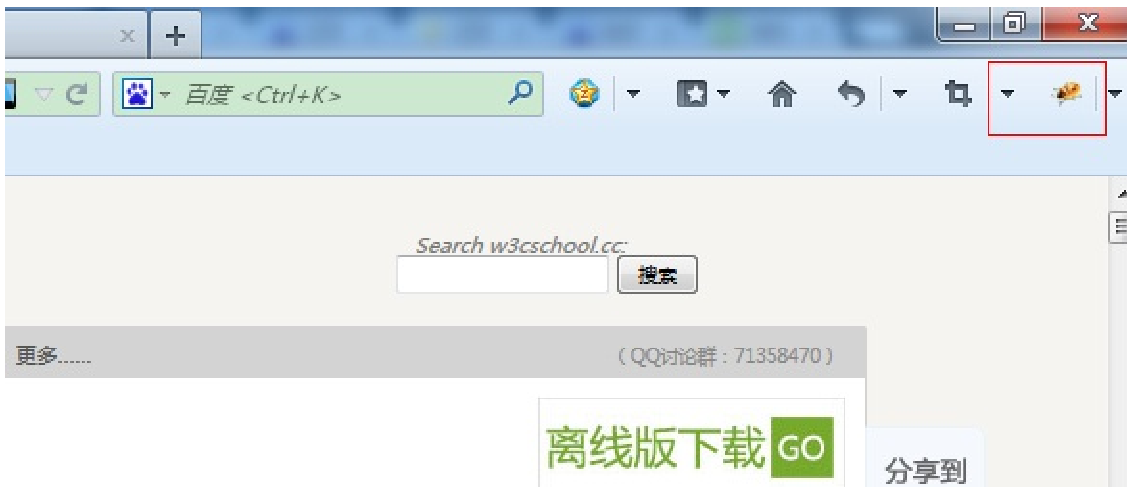
- 如何安装 Firebug。
- 使用 Firebug 检查和编辑 HTML。
- 使用 Firebug 检查和编辑 CSS。
- 使用 Firebug 调试 JavaScript。
- 在 Firebug 中动态执行 JavaScript。
- 使用 Firebug 记录 Javascript日志。
- 使用 Firebug 监控网络。

安装 Firebug

Firebug下载地址： https://addons.mozilla.org/en-US/firefox/addon/1843/

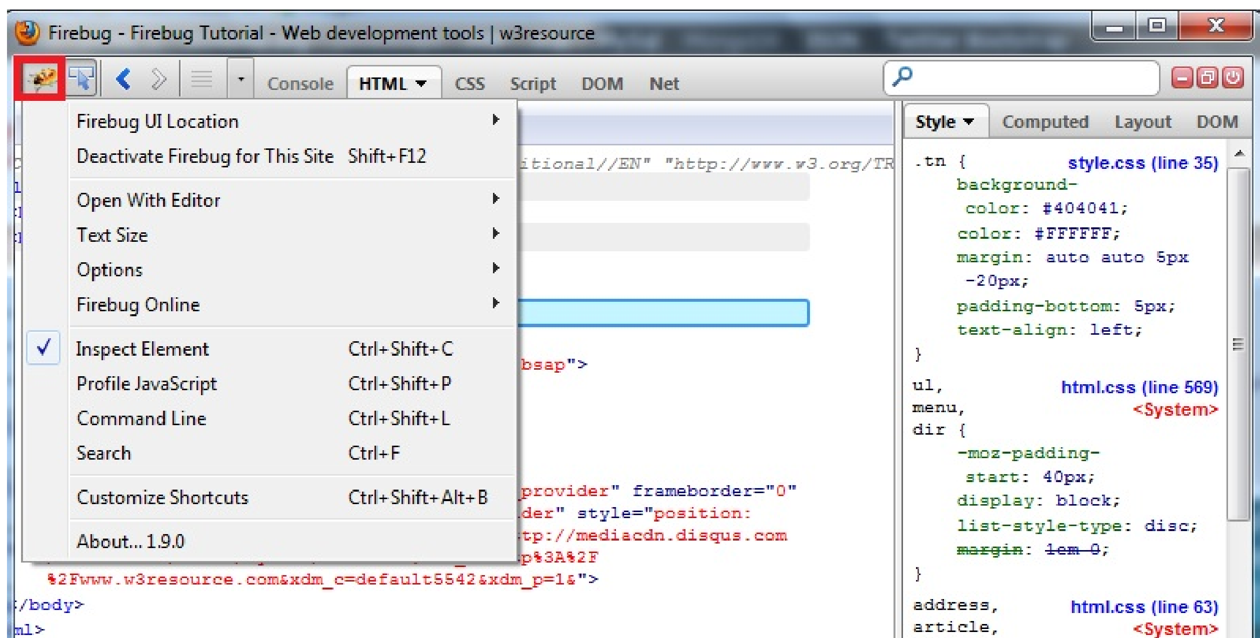
使用Firefox浏览器访问以上下载地址，打开页面后，点击下载按钮后，会有个弹窗页面，提示安装的目录，安装后重启你的Firefox浏览器。

重启后就可以在Firefox浏览器中看到Firebug的图标。点击Firebug图标(位于Firefox浏览器右上角)即可激活Firebug插件。

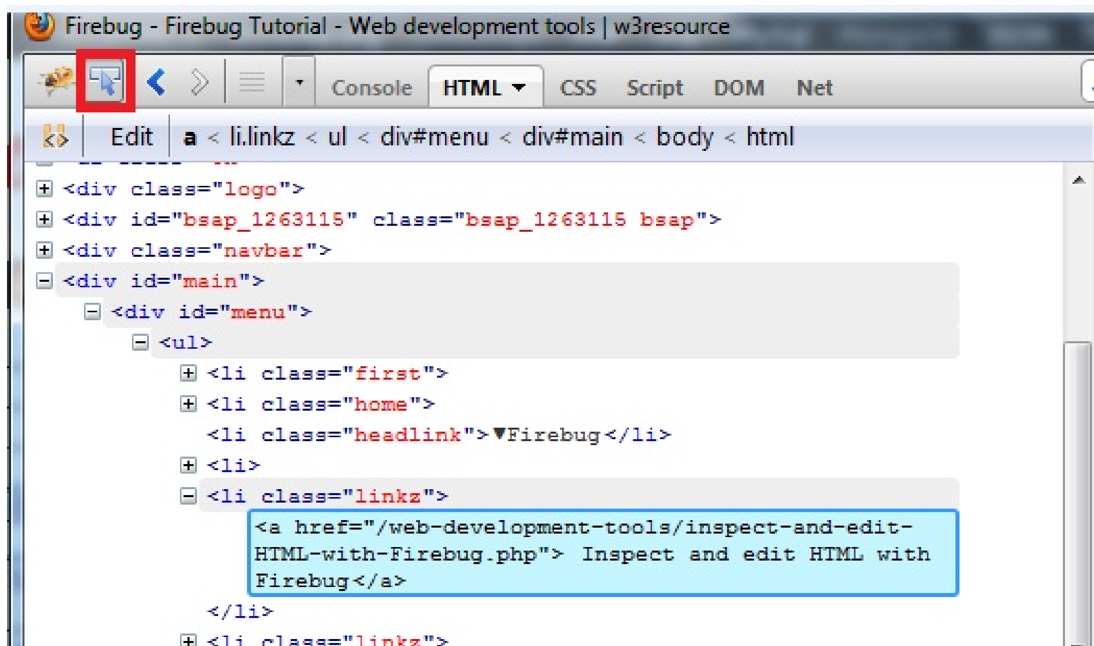


Firebug 组件

Firebug 选项



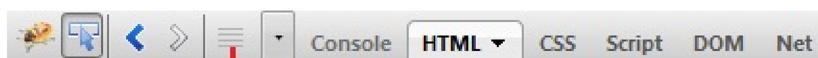
在页面上点击检查元素



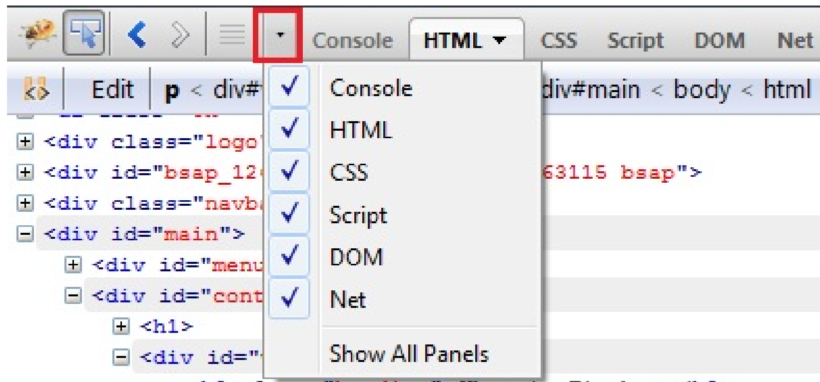
撤销与恢复



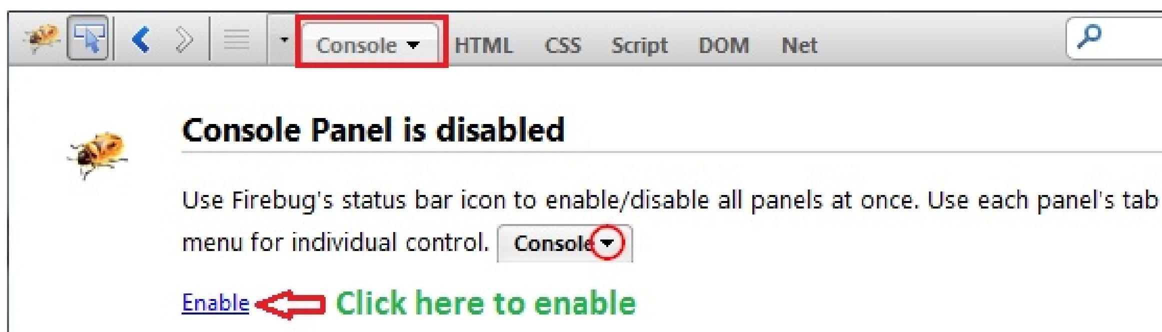
显示命令行



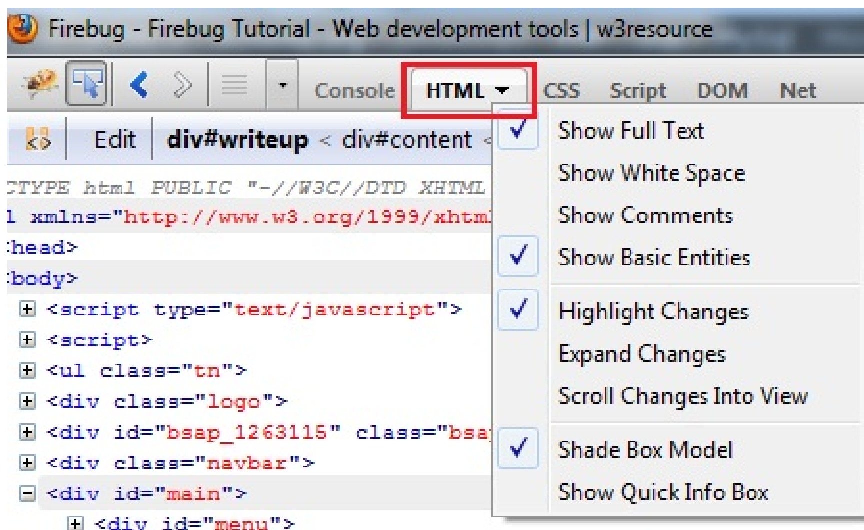
面板选择



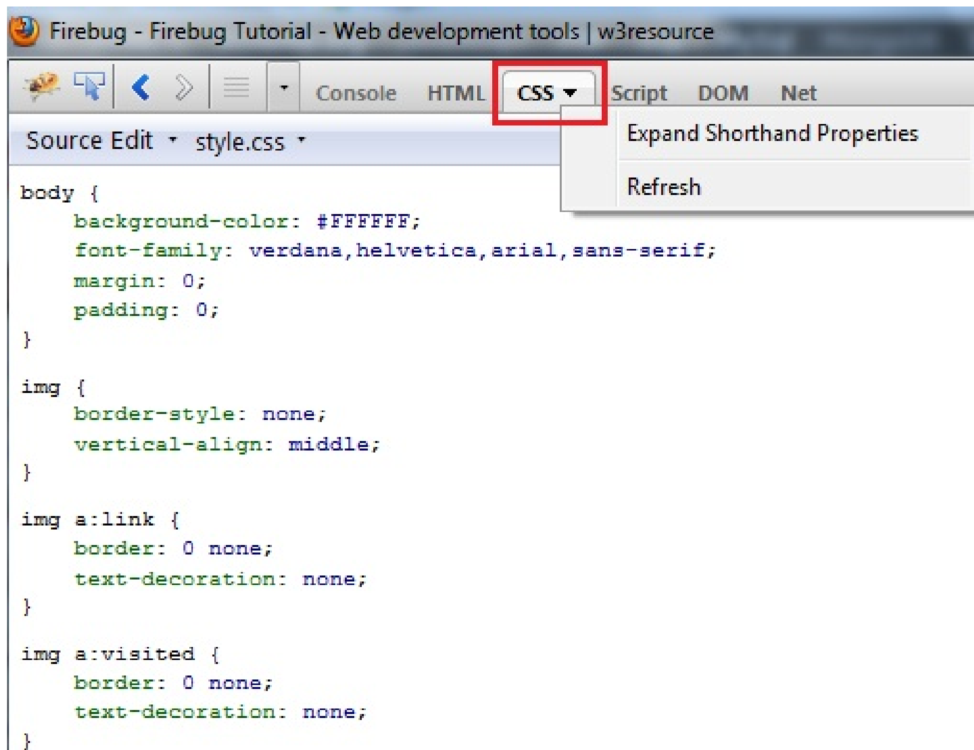
控制台



HTML 面板

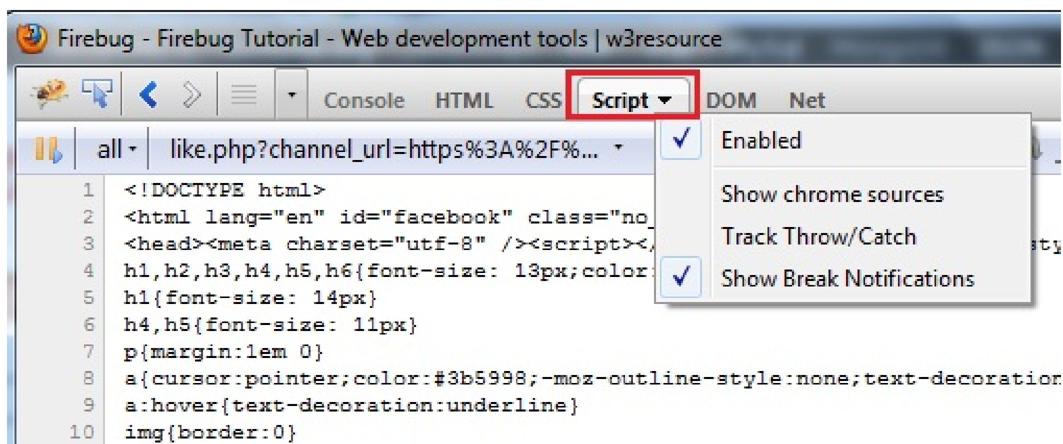


CSS 面板



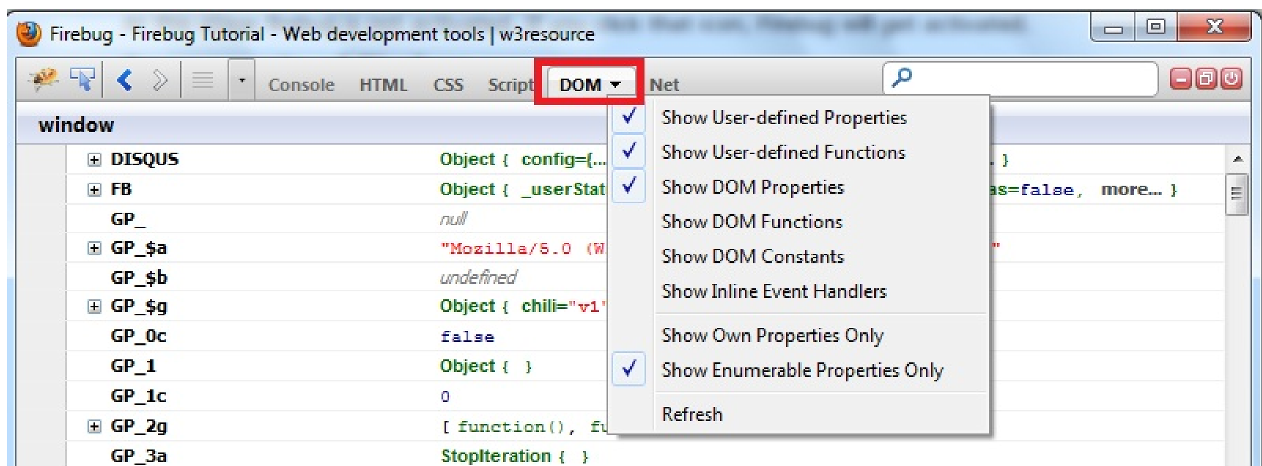
Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
Source Edit
style.css
Expand Shorthand Properties
Refresh
body {
background-color: #FFFFFF;
font-family: verdana,helvetica,arial,sans-serif;
margin: 0;
padding: 0;
}
img {
border-style: none;
vertical-align: middle;
}
img a:link {
border: 0 none;
text-decoration: none;
}
img a:visited {
border: 0 none;
text-decoration: none;
}

脚本面板



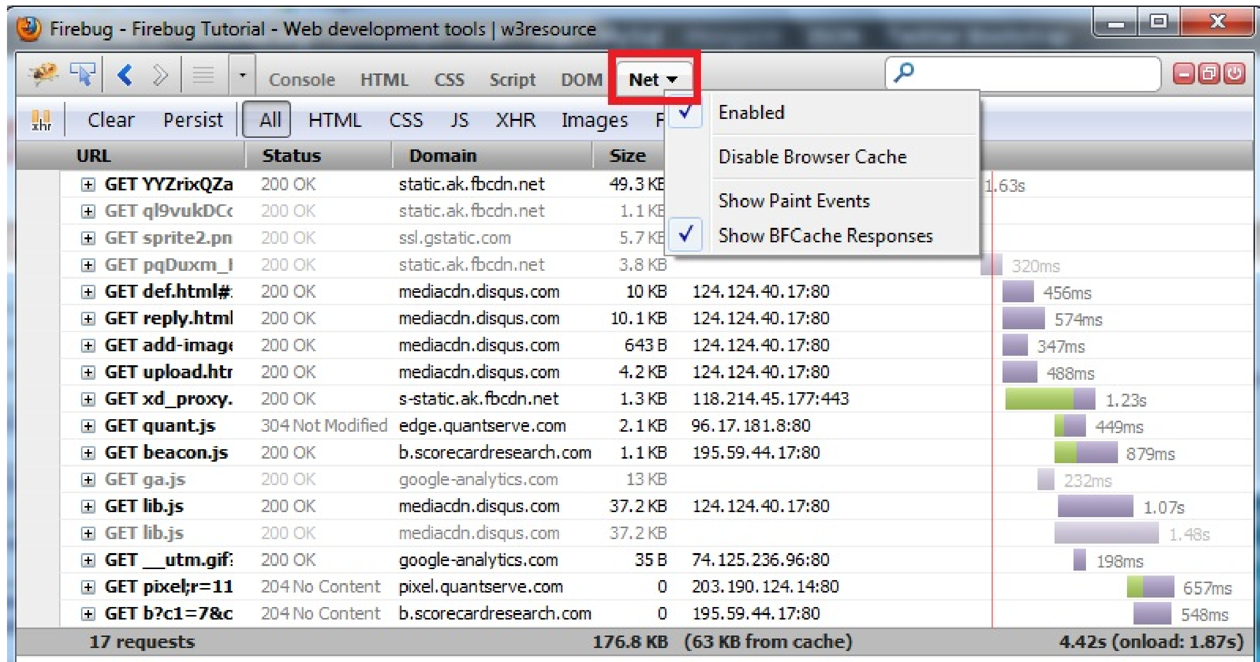
Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
all
like.php?channel_url=https%3A%2F%...
Enabled
Show chrome sources
Track Throw/Catch
Show Break Notifications
1 <!DOCTYPE html>
2 <html lang="en" id="facebook" class="no
3 <head><meta charset="utf-8" /><script><
4 h1,h2,h3,h4,h5,h6{font-size: 13px;color
5 h1{font-size: 14px}
6 h4,h5{font-size: 11px}
7 p{margin:1em 0}
8 a{cursor:pointer;color:#3b5998;-moz-outline-style:none;text-decoratior
9 a:hover{text-decoration:underline}
10 img{border:0}

DOM 面板



Firebug - Firebug Tutorial - Web development tools | w3resource
Console
HTML
CSS
Script
DOM
Net
window
Show User-defined Properties
Show User-defined Functions
Show DOM Properties
Show DOM Functions
Show DOM Constants
Show Inline Event Handlers
Show Own Properties Only
Show Enumerable Properties Only
Refresh
DISQUS
FB
GP_
GP_$a
GP_$b
GP_$g
GP_0c
GP_1
GP_1c
GP_2g
GP_3a
Object { config={...
Object { _userStat
null
"Mozilla/5.0 (W
undefined
Object { chili="v1'
false
Object { }
0
[function(), fu
StopIteration { }
more... }

网络面板 (用于网页测速和优化)



其他按钮



- 1 - 最小化Firebug。
- 2 - 在浏览器窗口安装Firebug。
- 3 - 停用Firebug。

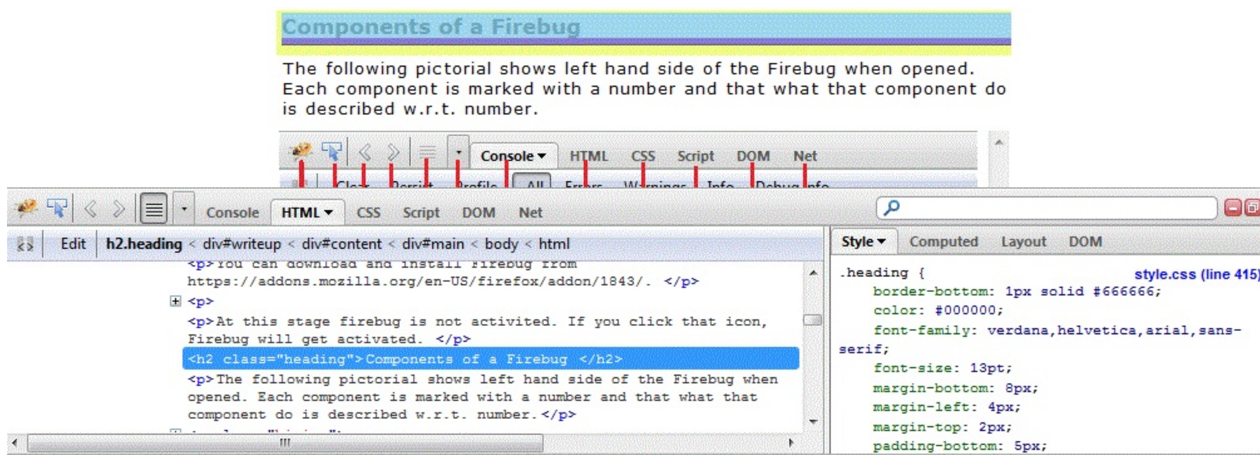
使用Firebug查看和编辑HTML和CSS

描述

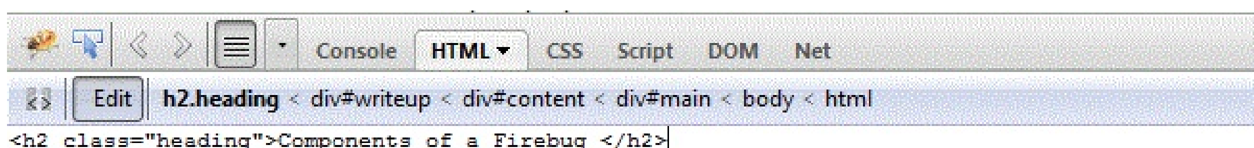
在本章节的教程中，我们将讨论如何使用Firebug查看和编辑HTML和CSS。

使用Firebug查看和编辑HTML

在你要查看的元素上右击鼠标然后点 **Inspect Element(查看元素)**.



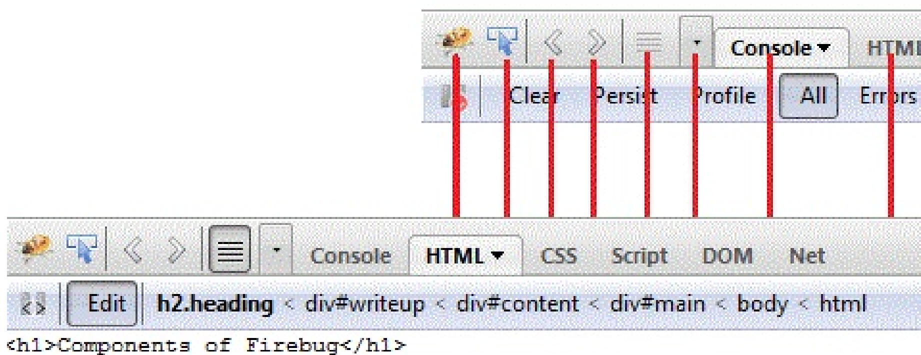
现在在Firebug窗口上点击**Edit**（编辑）。



修改代码 <h1>Components of Firebug</h1> .

Components of Firebug

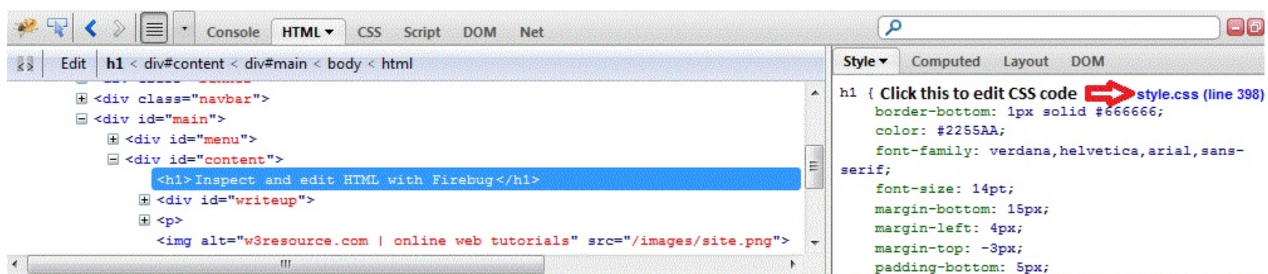
The following pictorial shows left t
Each component is marked with a
is described w.r.t. number.



你只要在Firebug窗口上修改代码就能实时的查看修改后代码在浏览器上的显示效果，你也可以复制代码到你的HTML文件中。

使用Firebug查看和编辑Css

鼠标右击你要查看的元素然后点击**Inspect Element(**查看元素**)**。如下图所示点击样式文件



通过修改代码为 **color:red;** 来修改标题颜色

修改完成后你可以马上看到修改后的效果。

现在你可以复制修改后的样式，取代原有的代码并保存，使之生效：


Home
▼Firebug Tutorials
Firebug Tutorials
Inspect and edit HTML with
Inspect and edit CSS with Firebug
Right click on the element which you want to inspec
Inspect Element.
Console
HTML
CSS
Script
DOM
Net
Edit style.css
width: 582px;
}
h1 {
border-bottom: 1px solid #666666;
color: red;
font-family: verdana,helvetica,arial,sans-serif;
font-size: 14pt;
margin-bottom: 15px;

使用 Firebug 调试 JavaScript

描述

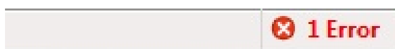
Firebug是一个非常强大的工具，可以帮助您发现代码发现错误的错误并解决错误。

在此我们使用Firebug来处理Javascript代码。

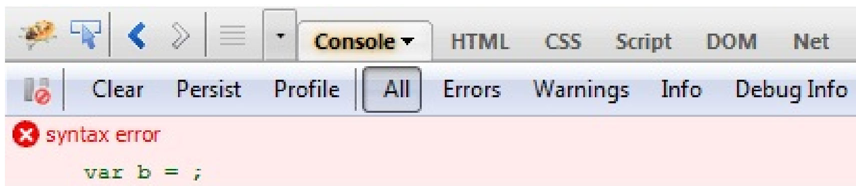
首先我们需要加载页面并打开Firebug。

有时候您需要重新载入页面。

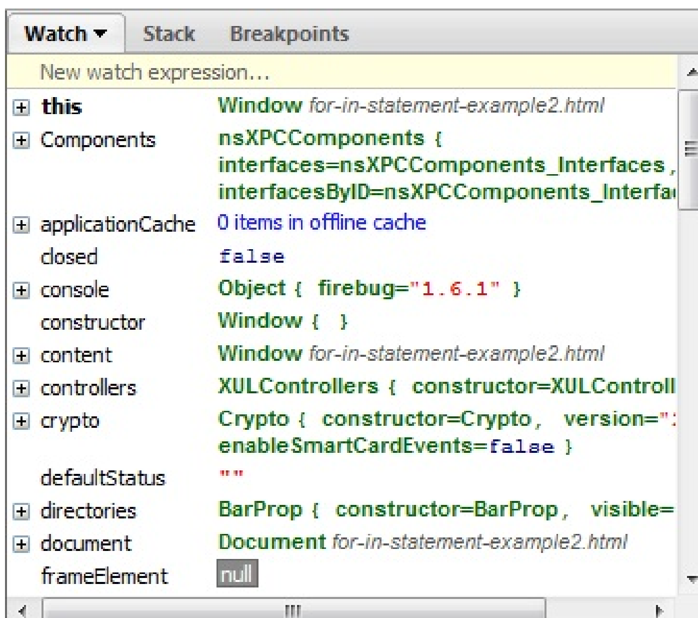
在状态栏的错误数



显示当前页面的错误

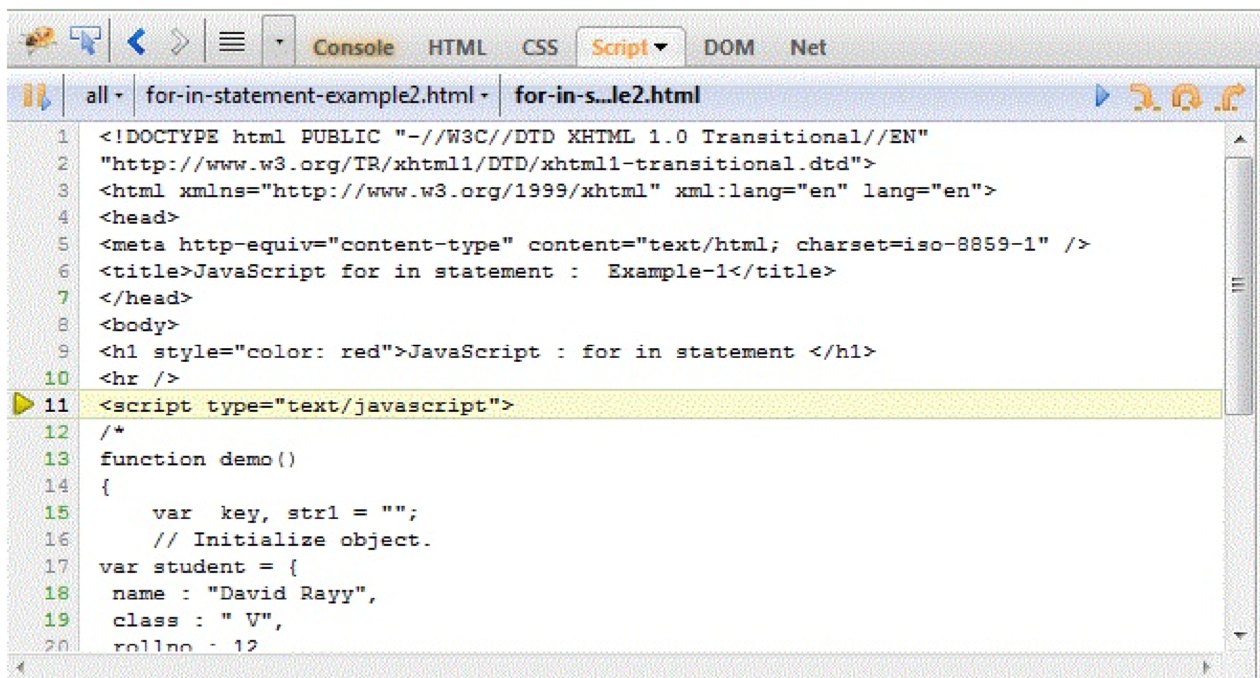


显示错误的详细信息



一步步调试代码

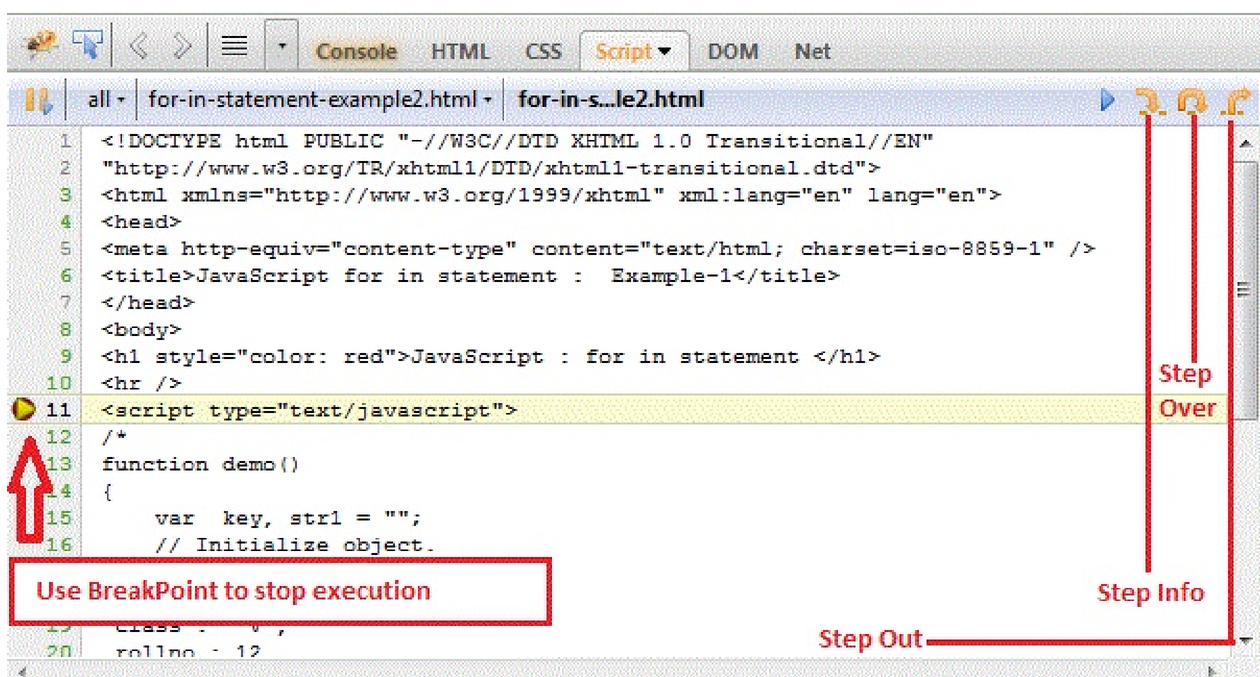
你可以一步步的执行代码。这对代码调试非常有用。



使用断点调试

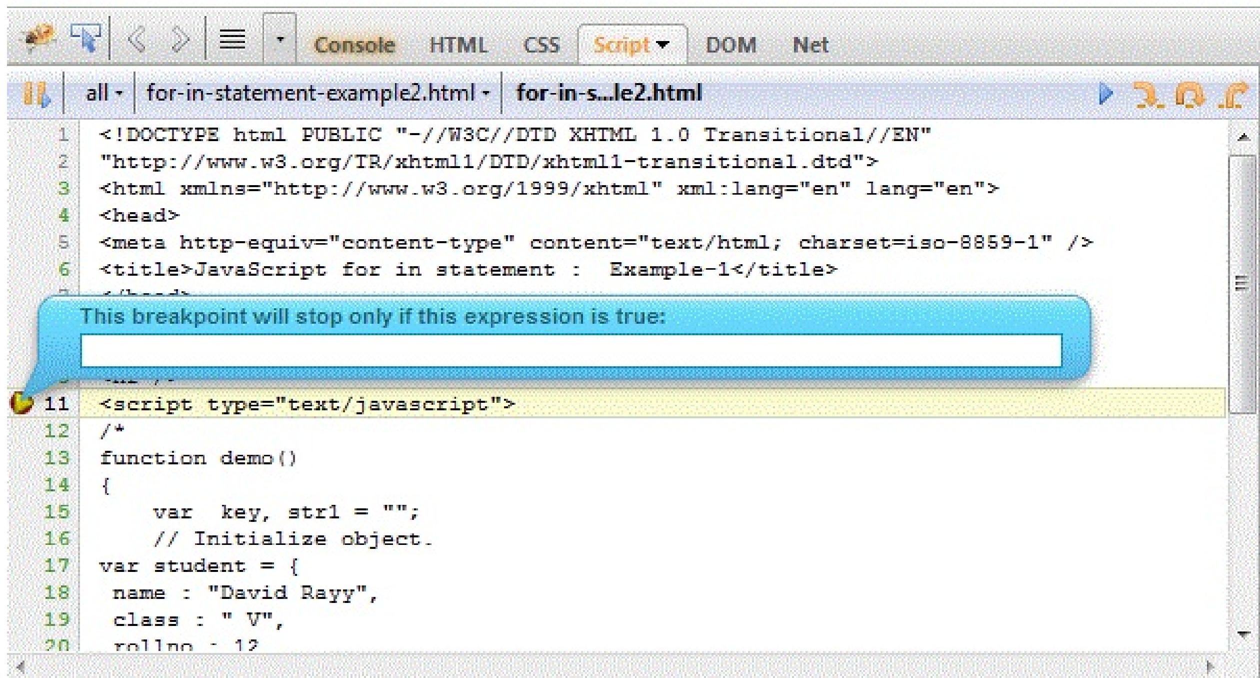
断点调试可以终止代码的执行，你可以通过指定代码范围来查看错误是不是在指定代码范围内。这对于 错误调试很有用。

如果你点击"Step over（单步跳过）" 按钮, Firebug 会更新所有变量直到你在右侧窗口中终止断点执行。



使用表达式让断点工作

你可以写一个表达式，在条件为真时，断点会停止代码的执行。



搜索

你可以使用快速搜索来查找代码中的关键字。



Firebug页面概况查看

使用Firebug的概况，你可以测试Web页面导致延迟加载的文件。

通过打开页面 Firebug > Console（控制台）> Profile（概况）。

你需要重新加载页面，然后点击"概况"查看页面载入情况。



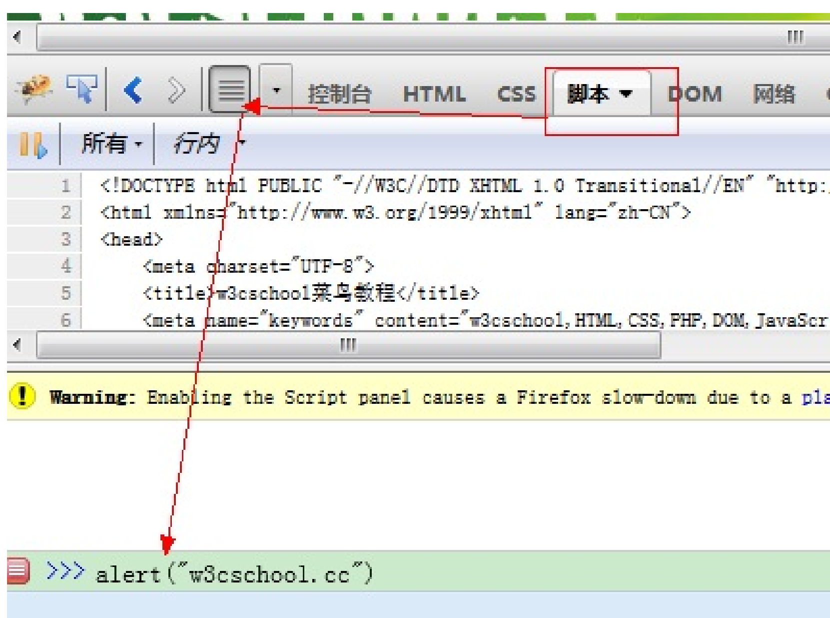
Firebug动态执行JavaScript

您可以使用Firebug来编写并实时执行一个JavaScript。

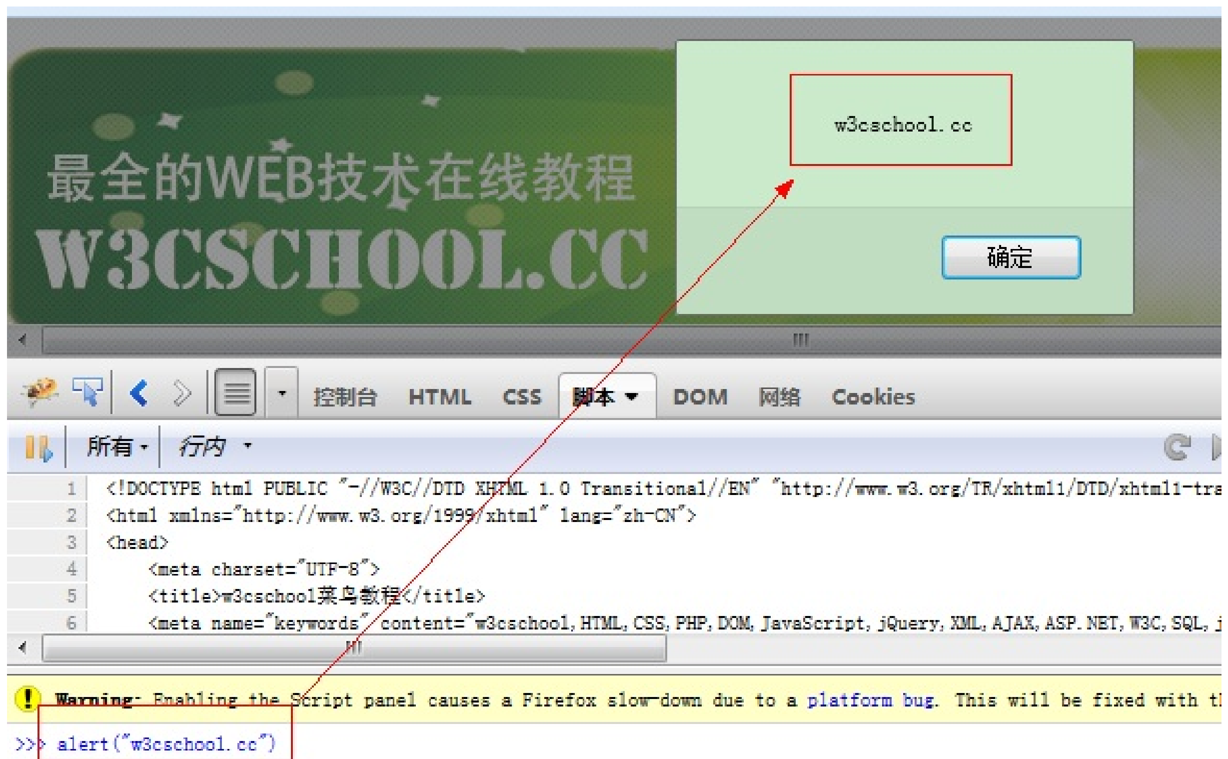
这是为了测试，并确保该脚本工作正常，这是将JavaScript代码部署在生产环境前的好方法。

打开Firebug > Script(脚本) > 显示命令行，代码如下所示：

```
alert("w3cschool.cc")
```



按Enter键后，马上你就能看到代码的输出。如图所示：


最全的WEB技术在线教程
W3CSCHOOL.CC
w3cschool.cc
确定
控制台 HTML CSS 脚本 DOM 网络 Cookies
所有 行内
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tra
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="zh-CN">
3 <head>
4 <meta charset="UTF-8">
5 <title>w3cschool菜鸟教程</title>
6 <meta name="keywords" content="w3cschool,HTML,CSS,PHP,DOM,JavaScript,jQuery,XML,AJAX,ASP.NET,W3C,SQL,j
Warning: Enabling the Script panel causes a Firefox slow-down due to a platform bug. This will be fixed with t
>>> alert("w3cschool.cc")

Firebug记录Javascript日志

你可以使用Firebug来生成日志。

这有助于我们调试web页面并发现页面的错误。

在Firefox浏览器中执行以下代码：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.o
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Firefox测试页面 - w3cschool菜鸟教程</title>
</head>
<body>
<script type="text/javascript">
var a = "w3cschool";
var b = ".cc";

document.write(a,b);
console.log(a + b);
</script>
</body>
</html>
```

使用Firefox浏览器打开以上代码文件firefox-test.html,执行结果及日志记录如下：



Firebug监控网络情况

Firebug可以监控网页中每个文件加载的时间。

打开Firebug。点击"网络"，然后确定已经启用，重新载入当前页面。Firebug显示如下所示：



免责声明

W3School提供的内容仅用于培训。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关。W3School简体中文版的所有内容仅供测试，对任何法律问题及风险不承担任何责任。